# COREOGRAFIA DE FLUXOS DE TRABALHO CIENTÍFICOS PARA COMPUTAÇÃO EM GRADE

ANDRÉ CARDOSO DE SOUZA
Orientador: Renato Antônio Celso Ferreira

# COREOGRAFIA DE FLUXOS DE TRABALHO CIENTÍFICOS PARA COMPUTAÇÃO EM GRADE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte, MG
Maio de 2009

de Souza, André Cardoso
Coreografia de Fluxos de Trabalho Científicos para Computação em Grade / André Cardoso de Souza. — Belo Horizonte, MG, 2009
xvi, 70 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Renato Antônio Celso Ferreira

1. Grades Computacionais. 2. Fluxos de Trabalho Científicos. 3. Gerenciamento de Fluxos de Trabalho. 4. Coreografia. 5. Arquitetura baseada em serviços. I. Título.

# [Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

# Agradecimentos

Em primeiro lugar, agradeço a Deus por se fazer presente em minha vida e me fazer alcançar o que minha mente nunca foi capaz de sonhar. Tu és o dono de tudo isso.

Agradeço aos meus pais e irmãos pelo incentivo e suporte que sempre me deram, pelo apoio e preocupação, pelo carinho e valorização.

Agradeço aos meus parentes e amigos, que dividiram comigo felicidades e preocupações, dentro e fora da universidade.

Agradeço ao meu orientador, Renato, por me dar oportunidade de relizar este trabalho e me guiar durante essa caminhada.

Agradeço a todos os demais que contribuíram para que eu concluísse mais essa etapa da minha vida.

# Resumo

As grades computacionais são um ambiente de computação distribuída amplamente utilizado. Utilizadas predominantemente em ambientes acadêmicos e de pesquisa científica, as grades impulsionaram a colaboração entre grupos de pesquisa muito distantes geograficamente e permitiram que instituições colaborassem em projetos de âmbito nacional, continental e global. Dentre as grades existentes, têm-se destacado aquelas cujo propósito é criar um ambiente para a pesquisa na área médica e/ou impulsioná-la. Um exemplo importante de tal tipo de grade é o caGrid, uma grade criada para impulsionar a pesquisa sobre o câncer. Ao redor de uma grade como o caGrid, comunidades científicas são formadas e novas demandas surgem. Uma das principais demandas atuais no caGrid é o desenvolvimento de uma ferramenta que auxilie os pesquisadores na utilização de fluxos de trabalho científicos, que são sequências complexas de tarefas que utilizam dados e serviços disponíveis na grade. A fim de suprir a necessidade da comunidade de usuários do *caGrid* de uma ferramenta de auxílio à utilização de fluxos de trabalho científicos, o autor desta dissertação desenvolveu o *caGrid choreography System* (caOS). Tal sistema permite que um cientista utilize uma descrição, em formato bem definido, de um fluxo de trabalho para criá-lo, executá-lo e monitorá-lo. Entre os resultados alcançados, destacam-se: o emprego de coreografia para execução de fluxos de trabalho científicos, o desenvolvimento de mecanismos de segurança integrados aos utilizados no caGrid e o fato de o caOS possuir uma arquitetura facilmente extensível.

# Abstract

Computational Grids are a broadly used distributed computing environment. They are used mainly in academic environments and scientific research and have improved the collaboration of research groups in projects deployed in national, continental and global levels. Among existing Grid environments, those seeking to create or improve an environment for research in medical sciences are being given a lot of attention. An important example of such a type of Grid environment is caGrid, which was created to improve scientific research on topics related to cancer. In a Grid environment like caGrid, communities are built and new needs arise. One of the main needs of caGrid is the development of a tool for helping researchers to use scientific workflows, which are complex sequences of tasks built on top of data and services available in the Grid. To meet the need of the community of users of caGrid for a tool to help in the use of scientific workflows, the author of this dissertation developed the caGrid choreography System (caOS). The developed system enables a scientist to create, execute and monitor a scientific workflow based on a description of the workflow in a well-defined format. The main results of this work are: the use of choreography for executing scientific workflows, the development of security mechanisms integrated to those already existing in caGrid and an easily extensible architecture of caOS.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

As grades computacionais, também conhecidas como *Grids*, são um ambiente de computação distribuída amplamente utilizado. Uma grade computacional é constituída de recursos computacionais, muitas vezes no estado-da-arte e geograficamente distribuídos, e a comunicação entre seus elementos é feita predominantemente através da Internet. Esse tipo de ambiente distribuído têm sido estudado, desenvolvido e utilizado há mais de uma década, baseando-se inicialmente no trabalho publicado por Ian Foster e Carl Kesselman em [Kesselman e Foster, 1998]. Atualmente, as grades são utilizadas nos mais diversos ramos da ciência, como: Computação, Física, Medicina, Geologia e Meteorologia, entre outros.

Utilizadas predominantemente em ambientes acadêmicos e de pesquisa científica, as grades impulsionaram a colaboração entre grupos de pesquisa muito distantes geograficamente e permitiram que instituições colaborassem em projetos de âmbito nacional, continental e global. Projetos como GriPhyN [Deelman et al., 2004], myGrid [Stevens et al., 2003] e caGrid [Saltz et al., 2006] foram criados tendo como núcleo a implementação de uma grade sobre a qual pesquisadores de diferentes instituições e áreas da ciência podem colaborar de várias formas. Dentre essas formas de colaboração, destacam-se: o compartilhamento de dados, a utilização de recursos computacionais de outras instituições para a realização de experimentos *in silico*, a disponibilização de serviços de análise de dados e a composição de serviços para execução de seqüências complexas de tarefas.

Dentre as grades existentes, têm-se destacado aquelas cujo propósito é criar um ambiente para a pesquisa na área médica ou impulsioná-la. O *caGrid* [Saltz et al., 2006] é uma grade criada para estimular e impulsionar a pesquisa sobre o câncer e tópicos relacionados na área de Biomedicina. O *caGrid* faz parte do *caBIG* (*cancer Biomedical Informatics Grid*) [Kakazu et al., 2004], um projeto que abrange todos os EUA. O *myGrid* [Stevens et al., 2003], criado no Reino Unido, também provê suporte para

pesquisa na área de Biomedicina utilizando grades computacionais. O projeto OBI-Grid [Konagaya, 2002], que abrange todo o Japão, também é voltado para pesquisa em Biomedicina.

Ao redor de uma grade como as citadas acima, comunidades científicas são formadas e novas demandas surgem. Dentro do caGrid [Saltz et al., 2006], por exemplo, cientistas executam suas tarefas diárias segundo processos formados, muitas vezes, por um conjunto de dados sobre o qual uma cadeia de tarefas são realizadas. Como exemplo, um cientista pode acessar um banco de dados de imagens médicas, selecionar um conjunto de imagens segundo certo critério, extrair características de tecidos e comparar tais características com outras imagens de referência. Muitas das atividades cotidianas de um cientista estão implementadas no caGrid, o que permite que sequências de tarefas realizadas por cientistas sejam executadas usando os recursos da grade. Tais sequências de tarefas são chamadas de fluxos de trabalho científicos. Um fluxo de trabalho científico, ou apenas fluxo de trabalho, pode ser representado por um grafo no qual os vértices, também chamados de atividades ou estágios, representam fontes de dados ou tarefas a serem executadas sobre os dados e as arestas representam as dependências de dados existentes na sequência de tarefas, conforme ilustrado na Figura 1.1. O aumento da utilização de fluxos de trabalho no *caGrid* tornou necessária a criação de uma ferramenta que auxiliasse os cientistas a utilizar fluxos de trabalho.

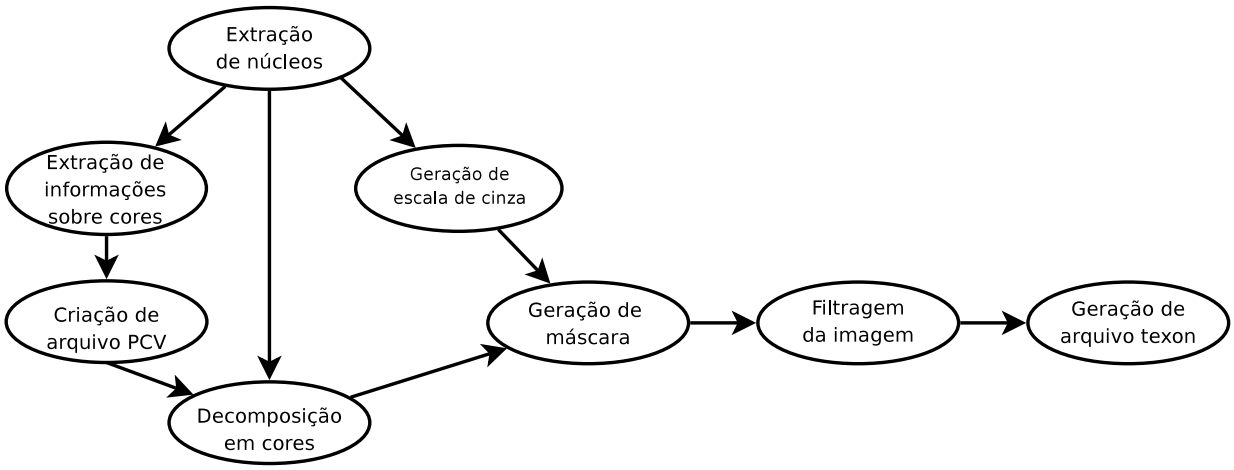


**Figura 1.1.** Diagrama de um fluxo de trabalho científico

Ferramentas de auxílio à utilização de fluxos de trabalho, também conhecidos como sistemas de gerenciamento de fluxos de trabalho, podem executar fluxos de trabalho segundo dois esquemas principais: orquestração e coreografia. O primeiro é semelhante à estratégia comumente utilizada por um cientista, que consiste, para cada tarefa, em: enviar os dados utilizados para execução, executar, recuperar os dados de saída e, quando for o caso, enviar os dados recuperados como entrada para execução de

outra tarefa. A utilização de um elemento central que controla a execução de cada tarefa e intermedia todas as transferências de dados define o esquema de orquestração [Ardissono et al., 2007], conforme ilustrado na Figura 1.2. No caso da orquestração, o elemento central é um ponto de contenção quando uma quantidade grande de dados é movimentada, pois cada dado é movimentado duas vezes, com exceção dos dados de entrada da primeira tarefa e os dados de saída da última. O segundo esquema, conhecido como coreografia, consiste em configurar cada tarefa para que seja executada independentemente e seus dados de saída sejam enviados para o destino sem atuação de um elemento central [Ardissono et al., 2007]. Ao executar um fluxo de trabalho segundo o esquema de coreografia, cada dado é movimentado apenas uma vez, o que evita as perdas de desempenho devido às transferências duplicadas de dados realizadas ao usar orquestração, e não há contenção no elemento central. A execução de um fluxo de trabalho segundo o esquema de coreografia é ilustrada na Figura 1.3.

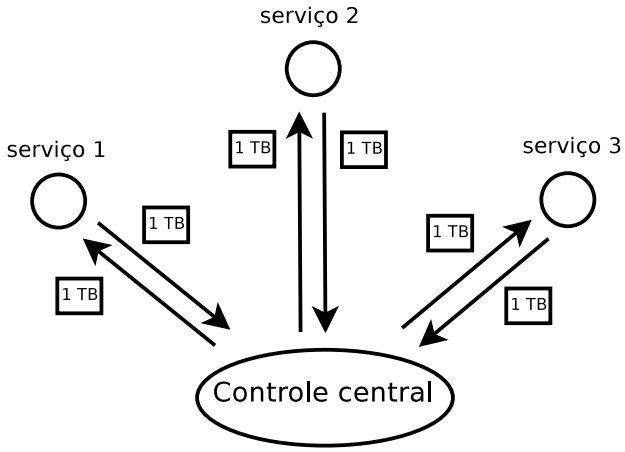


**Figura 1.2.** Execução de fluxos de trabalho segundo o esquema de orquestração

A utilização de serviços com acesso restrito também torna a segurança um fator importante para a utilização de fluxos de trabalho científicos. Frequentemente, serviços que disponibilizam dados confidenciais, como registros de pacientes de um hospital, restringem o acesso a um grupo de pesquisadores ou instituições para preservar a confidencialidade dos dados. A forma recomendada de acessar tais serviços é através da utilização de credenciais, que são chaves criptográficas que identificam um usuário numa grade [Butler et al., 2000]. Uma credencial é utilizada para criptografar a comunicação com o serviço e permitir ao serviço identificar o dono da credencial para, então, autorizar seu acesso. Os pesquisadores utilizam serviços com acesso restrito nas atividades de sua pesquisa e, por consequência, os utilizam, também, como parte de fluxos de trabalho científico. É necessário, então, que uma ferramenta de auxílio à

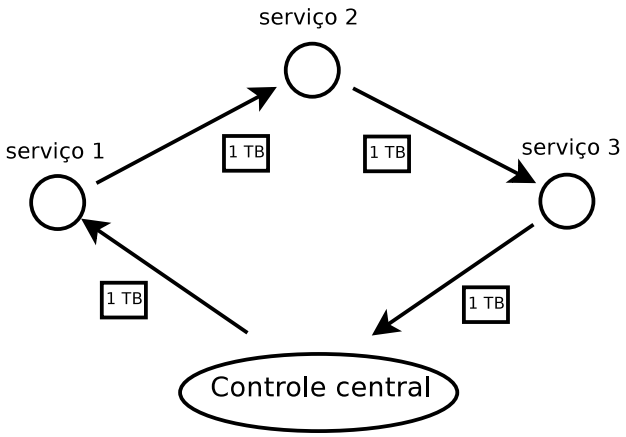


**Figura 1.3.** Execução de fluxos de trabalho segundo o esquema de coreografia

utilização de fluxos de trabalho cientificos seja capaz de acessar serviços seguros que porventura façam parte de um fluxo de trabalho.

A fim de suprir a necessidade da comunidade de usuários do *caGrid* de uma ferramenta de auxílio à utilização de fluxos de trabalho científicos, o autor desta dissertação desenvolveu o caGrid choreography System (caOS). Tal sistema permite a automatização da utilização de fluxos de trabalho através de uma descrição em formato bem definido. O caOS também é capaz de prover segurança no acesso a serviços e fontes de dados com acesso restrito, uma característica que não é encontrada em nenhum outra sistema de gerenciamento de fluxos de trabalho. Sistemas que auxiliam na criação, execução e monitoramento de fluxos de trabalho, como o caOS, são conhecidos como sistemas de gerenciamento de fluxos de trabalho [van der Aalst e van Hee, 2002]. O desenvolvimento do caOS favoreceu a interoperabilidade através da adoção de vários padrões, como: SOAP [W3C, 2007a], XPath [W3C, 2007c], X.509 [ITU-T, 2008], WS-Notification [OASIS, 2006a].

O restante deste capítulo é organizado como segue. A Seção 1.1 descreve o problema tratado neste trabalho. A Seção 1.2 discorre sobre os objetivos que foram propostos. Os principais resultados alcançados são descritos brevemente na Seção 1.3. A Seção 1.4 descreve a organização do texto desta dissertação.

## 1.1 Problema Abordado

Há vários aspectos motivadores do trabalho descrito na presente dissertação. Aqueles considerados mais relevantes serão abordados nesta seção. O problema abordado neste trabalho surgiu na comunidade de cientistas usuários de uma implementação de grade computacional chamada de caGrid [Saltz et al., 2006]. O caGrid foi criado como parte

de uma iniciativa de âmbito nacional do governo americano que tem como objetivo o desenvolvimento e aprimoramento constante de uma grade computacional que seja utilizada por cientistas da área médica em pesquisas sobre o câncer. O caGrid é utilizado por muitas instituições e, por isso, possui uma comunidade grande de usuários e também muitos serviços de acesso a dados e análise de dados. O caGrid é utilizado com frequência por cientistas que, após planejarem análises a serem feitas sobre um conjunto de dados, utilizam seus serviços para transformar dados, analisar os resultados e, possivelmente, submeter os dados resultantes a novas etapas de transformação até que informações consideradas relevantes sejam obtidas. Os serviços constituintes do caGrid são desenvolvidos como serviços Web, o que faz com que o caGrid seja considerado uma arquitetura orientada a serviços - na sigla em inglês, SOA: *Service Oriented Architecture*. Tal tipo de arquitetura têm sido utilizado por pesquisadores para criação de sequências complexas de tarefas conhecidas como fluxos de trabalho científicos. Tais fluxos de trabalho são processos executados com frequência por pesquisadores, mas seu desenvolvimento é, muitas vezes, prejudicado pela falta de especialistas em Computação que auxiliem os pesquisadores a criar fluxos de trabalho que realizem as atividades desejadas. Uma das abordagens para solucionar esse problema é desenvolver uma ferramenta que auxilie pesquisadores a criar, executar e monitorar fluxos de trabalho, diminuindo o esforço exigido de um pesquisador para criar um fluxo de trabalho que realize as atividades de sua pesquisa. Tais ferramentas são conhecidas como sistemas de gerenciamento de fluxos de trabalho científicos e vários destes já foram desenvolvidos (leia a Seção 3.3 e referências para detalhes).

A demanda por um sistema de gerenciamento de fluxos de trabalho que seja compatível com a arquitetura do caGrid e atenda às necessidades específicas da comunidade de pesquisadores usuários do caGrid é o problema abordado no trabalho descrito nesta dissertação. As necessidades específicas mais relevantes são: o emprego de coreografia para execução de um fluxo de trabalho e a integração com os mecanismos de segurança do caGrid, que incluem a utilização de credenciais para invocação de serviços.

## 1.2 Objetivos

O principal objetivo do trabalho descrito nesta dissertação é automatizar a utilização de fluxos de trabalho por parte dos pesquisadores pertencentes à comunidade do caGrid. Mais especificamente, o auxílio ao pesquisador deve ser feito através de um sistema de gerenciamento de fluxos de trabalho desenvolvido com este fim. Algumas características precisam estar presentes no sistema desenvolvido a fim de que este seja adequado ao uso no caGrid. Essas características serão detalhadas abaixo.

O primeiro objetivo do trabalho é criar um serviço capaz de instanciar cada elemento de um fluxo de trabalho, criando atividades executáveis e configurando a comunicação entre atividades que possuem dependências de dados. Define-se uma atividade do fluxo de trabalho, também chamada de estágio, como uma tarefa executada como parte de um fluxo de trabalho. Este primeiro serviço deve realizar a coreografia do fluxo de trabalho. Um aspecto importante da utilização da coreografia é a realização de transferências de dados entre estágios diretamente entre os mesmos, o que se opõe ao caso em que uma única entidade centraliza todas as transferências, conhecido como orquestração [Ardissono et al., 2007].

A capacidade de utilizar credenciais para configurar invocações de serviços seguros também deve estar presente, visto que muitos serviços de acesso a dados no caGrid possuem acesso restrito por mecanismos de segurança. A restrição de acesso pode existir, por exemplo, para controlar o acesso a dados confidenciais ou para mensurar a utilização, por um usuário, de um recurso computacional controlado por cotas. Com estas demandas atendidas, esse primeiro serviço é capaz de criar, configurar e executar um fluxo de trabalho.

Também é um objetivo importante do trabalho a criação de fluxos de trabalho a partir de uma descrição simples que possa ser construída com poucas dificuldades por pesquisadores. Tal descrição deve especificar: os serviços componentes, os requisitos de segurança de cada serviço, a localização em que cada estágio será executado (note que vários estágios podem executar o mesmo serviço com entradas diferentes), as credenciais utilizadas para acesso aos serviços seguros e as entradas/saídas do fluxo de trabalho. Um serviço deve, então, ser desenvolvido para receber a descrição citada e, utilizando-se das capacidades do primeiro serviço, criar e configurar cada elemento do fluxo de trabalho. A execução também deve ser iniciada por esse segundo serviço e este deve permitir que a evolução da execução seja monitorada.

Atendidas as demandas descritas nesta seção, pesquisadores usuários do caGrid serão capazes de criar e executar fluxos de trabalho com um esforço menor do que o exigido atualmente, mesmo sem o auxílio de especialistas em Computação. Tarefas realizadas com frequência, como transformações e análises de dados, serão facilitadas e agilizadas devido à automatização que pode ser feita através de um sistema de gerenciamento de fluxos de trabalho. A existência de um sistema de gerenciamento de fluxos de trabalho desenvolvido especificamente para a comunidade do caGrid também impulsiona o desenvolvimento de fluxos de trabalho sofisticados e a própria colaboração entre instituições, que poderão compor fluxos de trabalho a partir de serviços disponibilizados por várias instituições mais facilmente. Além disso, como o caGrid é baseado em padrões estabelecidos, a utilização do sistema não se restringe ao ambiente

do caGrid e permite que quaisquer serviços que obedeçam aos padrões internacionais sejam utilizados em fluxos de trabalho.

## 1.3 Principais Resultados

Dentre os resultados alcançados, aqueles considerados mais relevantes serão descritos brevemente nesta seção. O principal resultado do trabalho desenvolvido é a construção de um sistema de gerenciamento de fluxos de trabalho científicos, ao qual foi dado o nome de *caOS*. Tal resultado é de suma importância pois constitui a síntese de todos os objetivos descritos na Seção 1.2, ou seja, todas as demandas existentes na comunidade de usuários do caGrid quanto ao desenvolvimento e utilização de fluxos de trabalho e que foram incorporadas a este trabalho.

Destaca-se o emprego de coreografia para a execução de fluxos de trabalho. Ao implementar mecanismos para tal, foi possível evitar os custos adicionais devido a transferências duplicadas existentes quando estas são feitas por uma entidade central. Este é um aspecto inovador do trabalho apresentado aqui, já que transferências são feitas seguindo o modelo de orquestração na maior parte dos sistemas de gerenciamento de fluxos de trabalho mais utilizados [Oinn et al., 2004b, Taylor et al., 2003, Deelman et al., 2005] (veja mais detalhes na Seção 3.3).

O desenvolvimento de mecanismos de segurança integrados àqueles utilizados pelo caGrid também é um resultado importante. Muitos dos serviços Web utilizados no caGrid provêm acesso a dados de interesse para pesquisas médicas que só podem ser acessados por pesquisadores devidamente autorizados. No caGrid, por exemplo, dados de pacientes de hospitais são utilizados em pesquisas e a violação de sua confidencialidade pode gerar problemas jurídicos para os responsáveis pelos dados. Assim sendo, a incorporação dos mecanismos para acesso seguro a serviços restritos é fundamental para a criação de fluxos de trabalho no caGrid.

Outro resultado que merece ser mencionado é o fato de o caOS possuir uma arquitetura modular e com funcionalidades bem encapsuladas em serviços, o que permite que outras funcionalidades sejam acrescidas facilmente, também na forma de serviços. Também foi possível criar e executar fluxos de trabalho baseando-se numa descrição em formato XML [W3C, 2008], que é um formato amplamente conhecido não só por especialistas em computação, o que facilita a criação de fluxos de trabalho por parte de pesquisadores. Outro resultado obtido foi a capacidade de incorporar a um fluxo de trabalho qualquer serviço Web que permite ser invocado a partir de uma mensagem SOAP [W3C, 2007a], que é um padrão aberto.

Também merece destaque a utilização com sucesso de serviços do caGrid bas-

tante utilizados pela comunidade, como Data Transfer Service [Hastings, 2008], capaz de realizar transferências de arquivos entre estágios, e Credential Delegation Service [Langella, 2007], importante para a manipulação de credenciais realizada pelos mecanismos de segurança do caOS. É importante mencionar ainda que o custo adicional da utilização do caOS para instanciar e executar um fluxo de trabalho foi muito baixo quando comparado ao custo de uma execução que imita a estratégia utilizada pelo cientista para invocar diretamente cada estágio de um fluxo de trabalho, o que corresponde ao emprego de orquestração.

## 1.4 Organização da dissertação

Nesta seção, é descrita a organização do conteúdo desta dissertação nos capítulos seguintes. Conceitos necessários ao entendimento do trabalho descrito nesta dissertação, assim como uma visão geral da arquitetura do caOS, são apresentados no Capítulo 2. Trabalhos relacionados ao descrito nesta dissertação são apresentados e discutidos brevemente no Capítulo 3. O Capítulo 4 descreve detalhadamente os mecanismos implementados no caOS e sua arquitetura. O ambiente em que o caOS foi desenvolvido é descrito, abordando tanto aspectos gerenciais quanto operacionais, no Capítulo 5. O Capítulo 6 descreve os experimentos realizados utilizando o caOS, além de apresentar e analisar os resultados obtidos. Trabalhos que poderão suceder o apresentado aqui e as considerações finais do trabalho são apresentados no Capítulo 7. Por fim, são listadas as referências ao material utilizado como bibliografia.

# Capítulo 2

# Visão geral da arquitetura

Uma vez apresentadas as razões que motivaram o desenvolvimento deste trabalho e antes de prosseguir com a descrição detalhada do que foi feito, serão apresentadas, neste capítulo, algumas descrições e discussões que contribuirão para o entendimento do restante do trabalho. Um conjunto de conceitos imprescindíveis ao pleno entendimento do trabalho realizado é definido na Seção 2.1. O caGrid [Saltz et al., 2006], grade em cuja comunidade surgiu o problema motivador do trabalho descrito nesta dissertação, e seus serviços principais são descritos na Seção 2.2. Enfim, a arquitetura do caOS é descrita, em linhas gerais, na Seção 2.3.

## 2.1 Revisão de conceitos básicos

Nesta seção, são apresentados alguns conceitos importantes para o pleno entendimento do trabalho descrito nesta dissertação. Muitos dos conceitos apresentados correspondem a elementos que fazem parte do sistema desenvolvido ou dos quais o sistema desenvolvido depende. Será também discutida, de forma breve, a forma como cada conceito surgiu.

Um conceito bastante conhecido em Ciência da Computação é a execução de operações de forma remota. A forma mais simples de realizar tal tarefa consiste no modelo de chamada remota de procedimento (RPC) [Nelson, 1981], que descreve como um método invocado por um aplicativo pode ser executado em outro computador. Tal método remoto é definido por uma assinatura semelhante a qualquer procedimento local. Após o RPC, também surgiu um paradigma no qual é utilizado um par de aplicações conhecidas como cliente e servidor, através das quais é possível que o cliente execute uma operação específica no servidor. Com o surgimento da Web, o modelo RPC deu origem a um modelo no qual a comunicação é realizada sobre a Web e conjuntos de métodos são agrupados em serviços Web. A comunicação com um serviço Web é padronizada

pelo protocolo SOAP [W3C, 2007a], que define uma mensagem de invocação de um serviço como um arquivo XML [W3C, 2008]. Uma mensagem SOAP contém a assinatura do método a ser invocado no serviço Web e também contém os dados de entrada em formato XML. O serviço Web, ao receber uma mensagem SOAP e interpretá-la, executa a operação correspondente e envia os dados de saída ao chamador como uma mensagem SOAP. Um conjunto de serviços Web é hospedado num servidor Web, que também provê o ambiente de execução dos mesmos.

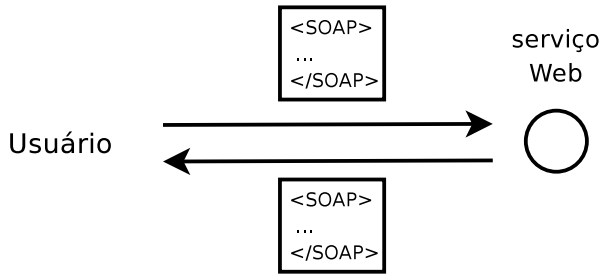


**Figura 2.1.** Invocação de um serviço Web usando mensagem SOAP

Serviços Web podem exigir que seus clientes o acessem de forma segura, principalmente quando é necessário restringir o acesso a dados confidenciais disponibilizados através do serviço ou controlar a utilização de recursos compartilhados por um grupo. Segundo o padrão *WS-Security* [OASIS, 2006c], uma invocação segura utiliza um par de chaves criptográficas pública/privada, chamado de credencial. O *WS-Security* assume que uma credencial obedece ao padrão X.509 [ITU-T, 2008]. Para invocar um serviço de forma segura, são seguidas as seguintes etapas: construção da mensagem SOAP, encriptação ou assinatura da mensagem, envio da mensagem para o serviço, decodificação da mensagem no serviço, autorização ao cliente para executar e execução. De forma semelhante, o retorno da invocação para o cliente é feito como segue: criação da mensagem SOAP pelo serviço, codificação ou assinatura da mensagem, envio ao cliente, decodificação da mensagem SOAP pelo cliente. A forma como é realizada uma invocação segura é ilustrada na Figura 2.2.

É possível a um usuário delegar a tarefa de invocação de um serviço a um terceiro. Isso é necessário, principalmente, quando deseja-se utilizar uma ferramenta de auxílio e a invocação de um serviço não é feita diretamente pelo usuário, mas sim pela ferramenta. No caso de um serviço sem restrições de acesso, não é necessária nenhuma ação adicional por parte do usuário. Porém, no caso em que o serviço a ser invocado exige acesso seguro, é necessário que o usuário transfira sua identidade e seus direitos para quem realizará a invocação. Isso é possível através da delegação de credencial, que consiste em criar uma nova credencial, chamada de credencial delegada, que é

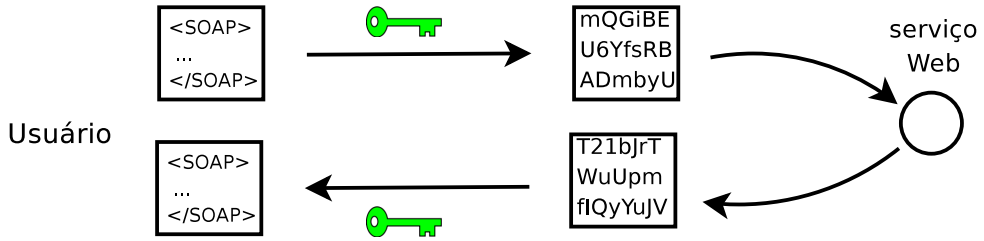


**Figura 2.2.** Invocação segura de um serviço Web

válida por um período restrito e permite que outro cliente seja identificado como se fosse o dono da credencial. Dessa forma, outro cliente pode acessar o serviço de forma segura e executá-lo em nome do dono da credencial. Neste trabalho, o processo de delegação de credenciais é realizado através do *caGrid Credential Delegation Service* (CDS) [Langella, 2007], um serviço do caGrid que permite a criação de credenciais delegadas. Conforme ilustrado na Figura 2.3, para que uma credencial pertencente a um usuário $A$ seja delegada a $B$, $A$ deve requisitar ao CDS a criação de uma credencial delegada associada a $B$ e informar o endereço do CDS a $B$. A credencial delegada deve então ser recuperada por $B$ junto ao CDS, que verifica a identidade de $B$ e entrega a credencial delegada por $A$. Feito isso, $B$ pode invocar um serviço cujo acesso só poderia ser feito por $A$. É importante ressaltar que a revogação da credencial de $A$ ou sua expiração invalida a credencial delegada a $B$, o que significa que $B$ não poderá mais agir em nome de $A$ se a credencial de $A$ tornar-se inválida. Outro aspecto importante é a restrição do uso da credencial para invocação de uma lista de serviços. A utilização de tal forma de restrição dificulta ações maliciosas por parte de quem recebe uma credencial delegada. Infelizmente, tal tipo de restrição não foi implementada no CDS e o Globus [Foster, 2005] também não possui suporte a tal tipo de restrição. Assim, usuários do caOS e serviços Web que se baseiam no Globus não poderão restringir o uso de uma credencial delegada a uma lista de serviços.

Uma chamada remota de procedimento é realizada, geralmente, de forma síncrona, ou seja, a execução do procedimento requisitado bloqueia a execução do procedimento que o invocou. Um problema recorrente em tais casos ocorre devido à longa duração que uma execução remota pode apresentar, o que leva a uma falha presumida pelo cliente. Uma interação assíncrona é útil para viabilizar, então, execuções remotas que duram um tempo muito longo. É comum, por exemplo, em serviços Web, a existência de operações que recebem como parâmetro um objeto que é utilizado para invocar um método informando o término da execução. Tal interação assíncrona pode ser implementada, entre serviços Web, sobre um mecanismo conhecido como notificação (cujo

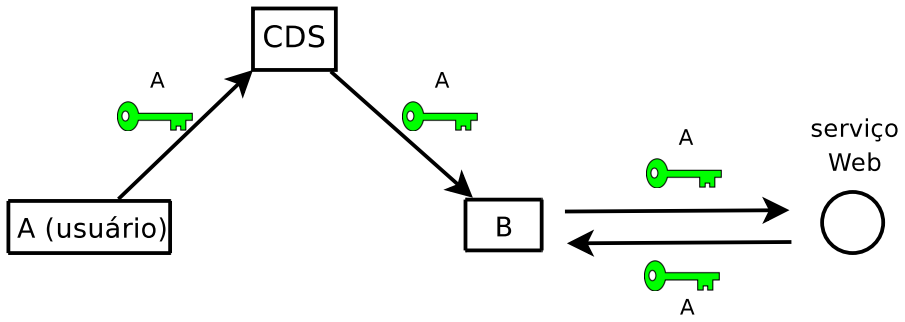


**Figura 2.3.** Delegação de uma credencial via *caGrid Credential Delegation Service*

padrão internacional é o *WS-Notification* [OASIS, 2006a]). Tal mecanismo tem como principal objetivo permitir a interação baseada em eventos entre serviços Web. É implementada definindo, num serviço Web, uma lista de tópicos de notificação. Outros serviços podem cadastrar-se no serviço como observadores de um dos tópicos, passando a receber uma mensagem sempre que uma mudança ocorrer no tópico observado. Com base em tal esquema de notificação, é possível implementar chamadas remotas assíncronas. Notificações também podem ser utilizadas para monitorar o estado de uma execução e detectar a ocorrência de falhas. Além disso, podem ser implementados mecanismos de garantia da qualidade de serviço através do monitoramento de tópicos que armazenem indicadores da qualidade de um serviço.

Além dos aplicativos que utilizam mecanismos baseados no paradigma cliente/servidor para executar, total ou parcialmente, utilizando recursos remotos, também existem outros paradigmas que utilizam recursos dessa forma. Um paradigma que difere dos demais apresentados acima é o paradigma de troca de mensagens [Gropp et al., 1996]. Tal paradigma consiste na utilização de mensagens para transferências de dados entre processos, os quais, por sua vez, processam tais dados e podem enviar resultados a outros processos. O paradigma de troca de mensagens é bastante apropriado para construção de aplicações que distribuem o processamento entre vários processos que executam o mesmo algoritmo. É bastante comum o desenvolvimento de aplicações que utilizam troca de mensagens sobre um ambiente com um conjunto de recursos homogêneos localizados numa única rede local ou que querem garantir a execução de seus serviços através de replicação massiva. Tal arquitetura é conhecida como aglomerado de computadores ou *cluster*. Por hospedar-se numa mesma rede local e, por consequência, num mesmo domínio administrativo, há poucas demandas de segurança, como autenticação e criptografia, sobre a execução de aplicações num aglomerado. Além de aplicações desenvolvidas sobre aglomerados de computadores, há ainda aplicações que utilizam recursos heterogêneos, de grande escala e distribuídos em

vários domínios administrativos. Tais recursos são, comumente, integrados num ambiente conhecido como grade computacional.

As grades computacionais têm sido utilizadas por pesquisadores para disponibilizar grandes quantidades de dados e executar processos para transformar ou analisar tais dados. Tais processos são constituídos por vários recursos disponíveis numa grade e, muitas vezes, são desenvolvidos em colaboração por várias instituições. O uso de tais processos tornou-se bastante difundido em diversas áreas da ciência e uma demanda por automação do uso dos mesmos surgiu naturalmente. Assim, foram desenvolvidos modelos para representação de tais processos. Tal representação foi padronizada [OASIS, 2007, W3C, 2005] constituindo-se, principalmente, das tarefas a serem realizadas, chamados de atividades ou estágios, e das dependências de dados e de controle entre as mesmas. Os processos desenvolvidos sobre grades computacionais e representados da forma descrita acima são chamados de fluxos de trabalho científicos. Além dos elementos citados, um fluxo de trabalho científico também pode possuir estruturas de controle, como: (i) execução condicional de um conjunto de estágios, (ii) execução repetitiva, (iii) execução obrigatoriamente em paralelo, (iv) execução obrigatoriamente em sequência, entre outros.

A popularização do uso dos fluxos de trabalho científicos - ou, simplesmente, fluxos de trabalho - fez com que os mesmos fossem utilizados por muitos pesquisadores que não são especialistas em Computação. Por isso, surgiu uma demanda por ferramentas de auxílio ao desenvolvimento e execução de fluxos de trabalho. Várias ferramentas de auxílio foram desenvolvidas com o objetivo de suprir tal demanda e ficaram conhecidas como sistemas de gerenciamento de fluxos de trabalho científicos [van der Aalst e van Hee, 2002] (na sigla em inglês, WFMS). Os WFMSs podem permitir a pesquisadores que não são especialistas em Computação: utilizar interfaces gráficas para criar um fluxo de trabalho a partir das atividades que o compõem, executar um fluxo de trabalho, monitorar uma execução, entre outras funcionalidades. O uso de WFMSs é bastante difundido atualmente e simplifica a utilização de fluxos de trabalho evitando que um pesquisador realize certas tarefas necessárias a tal utilização e que exigem conhecimento técnico em Computação.

Esforços têm sido empregados para a padronização do desenvolvimento e uso de WFMSs. Uma das iniciativas principais diz respeito à forma como um fluxo de trabalho é representado e executado [Peltz, 2003]. Os conceitos mais importantes em tal iniciativa são as definições de orquestração e coreografia. Na orquestração, é empregado um elemento centralizador que conhece o papel de todos os estágios e coordena as ações realizadas durante uma execução. Em particular, tal elemento centralizador atua como intermediário em todas as transferências de dados, o que torna o elemento

centralizador um ponto de contenção quando grandes quantidades de dados são transferidos. O emprego de orquestração para execução de um fluxo de trabalho é ilustrado na Figura 1.2. Em oposição à orquestração, na coreografia, cada estágio é informado de todas as ações pelas quais ele é responsável no fluxo de trabalho, tornando-se, então, capaz de atuar de forma independente e colaborar para a execução de todo o fluxo de trabalho. Em particular, cada estágio realiza as transferências necessárias para os estágios que dependem de seus dados de saída, evitando, assim, o surgimento de um ponto de contenção como o presente na orquestração de um fluxo de trabalho. O emprego de coreografia pode exigir a adição de elementos extras a um fluxo de trabalho a fim de fazer com que um estágio possa ser executado de forma independente e transferir seus dados de saída para outro estágio. A Figura 1.3 ilustra a execução de um fluxo de trabalho segundo o esquema definido como coreografia.

Na próxima seção, serão apresentadas as características do caGrid e seus serviços principais. Em seguida, a arquitetura do sistema de gerenciamento de fluxos de trabalho desenvolvido é apresentada em linhas gerais.

## 2.2 caGrid

Esta seção descreve o caGrid [Saltz et al., 2006], grade em cuja comunidade surgiu o problema abordado nesta dissertação. Seus principais serviços também serão descritos brevemente.

O caGrid [Saltz et al., 2006] é uma grade computacional criada para estimular a pesquisa sobre o câncer e tópicos relacionados na área de Biomedicina. Sua arquitetura é baseada em serviços Web e em padrões como *Simple Object Access Protocol* (SOAP) [W3C, 2007a], XPath [W3C, 2007c], *Web Services Resource Framework* (WSRF) [OASIS, 2006b], WS-Notification [OASIS, 2006a], X.509 [ITU-T, 2008], entre outros. Possui serviços que provêm funcionalidades como: registro e descoberta de serviços, padronização de estruturas de dados que representam conceitos, segurança. Os principais serviços que participam no provimento das funcionalidades serão descritos abaixo. Além destes, serão descritos os tipos de serviços criados por usuários.

O registro e a descoberta de serviços são possíveis através do serviço de indexação [Covitz et al., 2003]. É recomendado que cada serviço do caGrid registre-se no serviço de indexação a fim de que o conjunto de serviços conhecidos seja o mais abrangente possível. No momento em que um serviço se registra, são armazenadas informações como: localização do serviço, instituição hospedeira, etc. O índice pode ser acessado utilizando consultas XPath [W3C, 2007c], o que permite recuperar listas de serviços cuja descrição satisfaz um conjunto de critérios expresso na consulta XPath.

Outra funcionalidade importante no caGrid é a definição de conceitos do domínio de aplicações e a padronização das estruturas de dados que os representam. Tal funcionalidade permite que serviços que utilizam os mesmos conceitos, como o conceito de uma lâmina de microscópio, sejam interoperáveis através da utilização da mesma representação do conceito utilizado, como imagens digitalizadas para representar uma lâmina de microscópio. As definições de conceitos dos domínios das aplicações do caGrid são armazenadas no *Enterprise Vocabulary Service* (EVS) [Covitz et al., 2003]. Tais conceitos são referenciados na forma de anotações em definições de classes UML correspondentes às estruturas de dados que representam conceitos. Tais definições são armazenadas no serviço chamado de *cancer Data Standards Repository* (caDSR) [Covitz et al., 2003]. Assim, é possível aos serviços desenvolvidos no caGrid utilizar uma única representação de conceitos relevantes para várias aplicações e, por consequência, serem interoperáveis.

Os requisitos de segurança do caGrid são satisfeitos por uma infraestrutura chamada de *Grid Authentication and Authorization with Reliably Distributed Services* (GAARDS) [Langella et al., 2007]. Dentre as funcionalidades providas pelo GAARDS, duas das mais importantes são: o gerenciamento de contas de usuários e a delegação de credenciais. O Dorian [Langella et al., 2006] é um serviço que permite a criação de contas de usuário e o gerenciamento de contas registradas no caGrid. O Dorian é capaz de criar credenciais para usuários do caGrid e armazená-las para que os usuários a recuperem depois, através de uma autenticação baseada em nome de usuário e senha. A delegação de credenciais é implementada pelo *caGrid Credential Delegation Service* (CDS) [Langella, 2007]. O conceito de delegação de credenciais, bem como a forma com que o CDS é utilizado são descritas na Seção 2.1.

No caGrid, serviços desenvolvidos pelos usuários podem ser classificados em duas categorias: serviços de dados e serviços de análise. Serviços cuja função é proporcionar o acesso a um conjunto de dados são chamados de serviços de dados. Um requisito destes serviços é a representação dos dados conforme a definição armazenada pelo caDSR [Covitz et al., 2003]. Além disso, é necessário que a recuperação de dados seja realizada utilizando CQL [caGrid, 2007], uma linguagem de consulta desenvolvida para uso no caGrid. A segunda categoria de serviços desenvolvidos pelos usuários são os serviços de análise. Tais serviços proporcionam a execução de procedimentos de análise de dados. Os procedimentos precisam ser acessíveis através de interfaces orientadas a objetos e fortemente tipadas.

## 2.3 Visão geral da arquitetura

Nesta seção, os serviços que fazem parte da arquitetura do caOS são descritos resumidamente. Os mais importantes mecanismos utilizados são citados. Alguns detalhes sobre a implementação dos serviços também são mencionados.

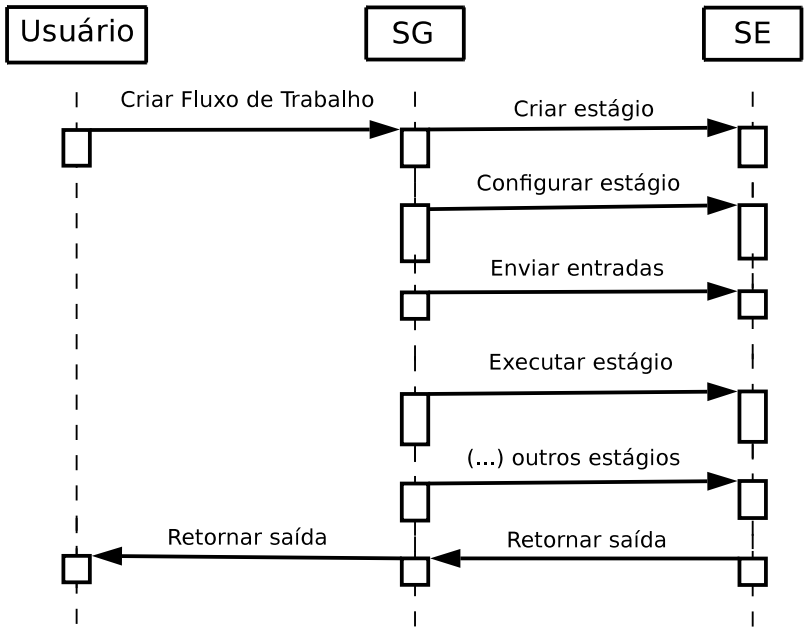


**Figura 2.4.** Interação entre os serviços do caOS para criação de um fluxo de trabalho

A arquitetura do caOS é composta por dois principais: o responsável pelo serviço de suporte à execução de fluxos de trabalho (SE) e o responsável pelo serviço de gerenciamento (SG). O serviço de suporte à execução é responsável por instanciar elementos de um fluxo de trabalho e controlar a execução de cada um destes. O SE é capaz de criar estágios, configurá-los e executá-los, mas não é capaz de controlar um fluxo de trabalho como um todo. O serviço de gerenciamento de fluxos de trabalho é responsável por instanciar um fluxo de trabalho completo a partir de uma descrição de seus componentes, entradas e saídas. Um estágio é criado através de uma chamada ao SE e a localização em que o mesmo é instanciado faz parte da descrição do fluxo de trabalho. A criação de um fluxo de trabalho é ilustrada na Figura 2.4. Note que a interação do usuário com o caOS é assíncrona e, por isso, permite que fluxos de trabalho longos sejam executados sem prejuízo da comunicação com o usuário, que poderia ser perdida no caso de uma comunicação síncrona devido ao *timeout* da execução das operações.

Os componentes do caOS foram implementados como dois Web com o auxílio do Introduce [Hastings et al., 2007], uma aplicação que utiliza as bibliotecas do Globus [Foster, 2005] para gerar serviços Web que implementam os mecanismos de comu-

nicação necessários ao seu uso no caGrid [Saltz et al., 2006] e possuam protótipos para operações definidas pelo usuário que constarão da interface do serviço. Geralmente, as operações dos serviços Web não possuem estado, sendo executadas apenas com base nos valores de seus parâmetros de entrada. No entanto, com o desenvolvimento de serviços responsáveis por múltiplos recursos, a gerência de tais recursos foi padronizada de forma semelhante ao padrão de projeto *factory* [Gamma et al., 1994], segundo o qual instâncias de um objeto não são criadas diretamente, mas através de uma classe fábrica. Para um serviço Web, o padrão foi chamado de WSRF [OASIS, 2006b] e define como criar um contexto do serviço (um recurso com estado) para acessar um recurso específico. No caOS, os elementos de um fluxo de trabalho foram representados por contextos dentro de cada serviço Web, conforme ilustra a Figura 2.5. Isso foi possível porque o Introduce pode tornar um serviço Web capaz de instanciar tais tipos de recursos.

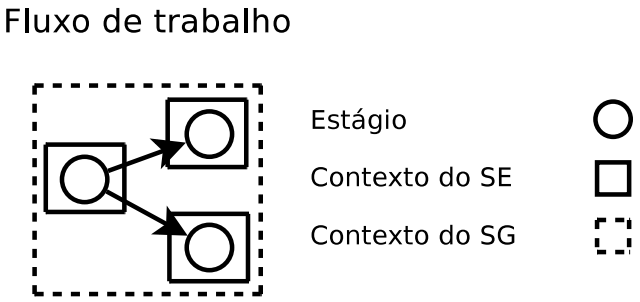


**Figura 2.5.** Fluxos de trabalho são controlados pelo caOS através de contextos dos serviços Web componentes

Entre os mecanismos implementados pelo caOS merecem destaque a coreografia de fluxos de trabalho e a manipulação de credenciais. A coreografia foi implementada através de um dos contextos do serviço de suporte à execução. Tal contexto é responsável pela execução de um estágio e também pode ser configurado para realizar transferências de dados a partir de um estágio do fluxo de trabalho sem atuação de uma entidade centralizadora. O caOS também implementa mecanismos que utilizam credenciais para invocar serviços de forma segura. É necessário que o usuário delegue uma credencial ao caOS, através do serviço de delegação de credenciais do caGrid [Langella, 2007], e então o sistema utiliza a credencial delegada para invocar os serviços que exigem acesso seguro.

## 2.4 Sumário

Neste capítulo, foram apresentados vários conceitos necessários ao entendimento do trabalho descrito nesta dissertação. Também foi apresentada, de forma resumida, as características e os serviços principais do caGrid [Saltz et al., 2006]. Por fim, foi descrita a arquitetura do sistema desenvolvido para abordar o problema de desenvolver e executar com facilidade fluxos de trabalho científicos no ambiente do caGrid. No Capítulo 3, serão apresentados vários trabalhos cujo tema se relaciona ao abordado nesta dissertação. A solução desenvolvida para o problema abordado, o sistema chamado de caOS, é descrita em detalhes no Capítulo 4.

# Capítulo 3

# Trabalhos relacionados

Existem muitos trabalhos que são relacionados de alguma forma ao descrito nesta dissertação. Alguns deles serão descritos neste capítulo. Exemplos de grades computacionais são descritos na Seção 3.1. Em seguida, a Seção 3.2 apresenta algumas linguagens utilizadas para descrever fluxos de trabalho científicos. A Seção 3.3 descreve vários sistemas de gerenciamento de fluxos de trabalho.

## 3.1 Grades computacionais

O trabalho apresentado nesta dissertação baseia-se, em grande parte, nos conceitos e na tecnologia das grades computacionais. Por isso, algumas delas serão descritas nesta seção.

O caGrid [Saltz et al., 2006] é uma grade computacional criada para estimular a pesquisa sobre o câncer e tópicos relacionados na área de Biomedicina. Sua arquitetura é baseada em serviços Web e inclui componentes capazes de prover autenticação através de credenciais X.509 [ITU-T, 2008] e transferências de arquivos. Os serviços do caGrid obedecem a vários padrões abertos, como SOAP [W3C, 2007a], WSDL [W3C, 2007b] e WSRF [OASIS, 2006b].

O projeto GriPhyN [Deelman et al., 2004] têm desenvolvido uma grade com o objetivo de gerenciar grandes quantidades de dados, que chegam a *petabytes*, para a realização de experimentos na área de Física. Foi construído com auxílio das bibliotecas do Globus [Foster, 2005] e também apresenta arquitetura baseada em serviços Web. Resultados intermediários e demais tipos de dados podem ser armazenados, manipulados e movimentados através de vários serviços implementados na grade. Os grandes experimentos que guiam o desenvolvimento do GriPhyN estudam Astronomia, detecção de ondas gravitacionais de Einstein e partículas da alta energia.

O *National Grid Service* (NGS) [Geddes, 2006] é a principal grade computacional

do Reino Unido e foi construído para prover o acesso a recursos computacionais e dados a pesquisadores daquele país. É utilizado em pesquisas sobre Bioquímica, Física, Medicina, Demografia, entre outros. Semelhantemente a outras grades citadas, os serviços que compõem a grade são serviços Web, assim como as interfaces dos recursos computacionais utilizados por pesquisadores em seus experimentos.

TeraGrid [Cattlet, 2005] é uma grade computacional criada para pesquisa científica em áreas diversas. Utiliza as bibliotecas do Globus [Foster, 2005] para construir a infra-estrutura que disponibiliza a seus usuários. Podem ser executados sobre o TeraGrid vários tipos de aplicações, como: serviços Web, tarefas GRAM [Feller et al., 2007] aplicativos MPICH [Karonis et al., 2003] e PBS [Henderson, 1995]. Transferências de dados podem ser feitas entre componentes da grade através do GridFTP [Allcock et al., 2002]. O TeraGrid têm sido utilizado por pesquisadores das áreas de Biomedicina, Astronomia, Química, entre outros.

O *Enabling Grids for e-Science* (EGEE) é uma grade computacional iniciada na Europa e utilizada em âmbito global. Pesquisadores que a utilizam são oriundos dos mais diversos ramos da ciência, como: Astronomia, Astrofísica, Geociências e Ciência da Computação. Sua infra-estrutura baseia-se no gLite [Laure et al., 2006], um arcabouço para criação de aplicações em grade que possui uma arquitetura baseada em serviços. O EGEE utiliza também padrões adotados internacionalmente a fim de garantir a interoperabilidade entre os diversos recursos constituintes da grade. Entre as funcionalidades presentes no EGEE destacam-se mecanismos de segurança baseados em credenciais X.509 [ITU-T, 2008], assim como armazenamento e transferência de arquivos.

## 3.2 Linguagens para descrição de fluxos de trabalho científicos

Existem várias linguagens criadas com o objetivo de descrever fluxos de trabalho científicos. Neste trabalho, foi criado e utilizado um formato para descrever fluxos de trabalho científicos, que é descrito na Seção 4.3.1. O formato para descrição utilizado é o equivalente à serialização em formato XML do objeto descritor exigido pelo SG para criação de um fluxo de trabalho. Apesar de não ter sido criada uma linguagem para descrever um fluxo de trabalho, existem várias linguagens que poderiam ter sido utilizadas. A utilização do formato adotado permite que as linguagens apresentadas sejam utilizadas para acessar o caOS, desde que sejam implementados transformações de tais linguagens par ao formato utilizado pelo caOS. Nos parágrafos seguintes, algumas linguagens dentre as mais utilizadas para descrever fluxos de trabalho científicos

são descritas.

WS-BPEL [OASIS, 2007] é uma linguagem que permite descrever processos de negócios que representam fluxos de trabalho científicos e assume o emprego de orquestração. Tal linguagem é um padrão já estabelecido que baseia-se na linguagem XML e permite que sejam descritos vários elementos de um fluxo de trabalho, como: serviços a serem invocados, transferências de dados, parâmetros de entrada e saída. Também é possível utilizar atividades - como execução condicional, sequência de operações, operações paralelas, escolha dentre um conjunto de operações e execução repetitiva - para descrever a forma que um fluxo de trabalho deve ser executado. Mecanismos para tratamento de falhas também podem ser descritos num documento WS-BPEL. Algumas ferramentas foram construídas para executar fluxos de trabalho, tanto científicos como de natureza comercial a partir de documentos em conformidade com a especificação WS-BPEL.

A linguagem Scufl [Oinn et al., 2004a] é utilizada pelo Taverna [Oinn et al., 2004b] como padrão para descrição de fluxos de trabalho. Sua especificação simples de um fluxo de trabalho permite que descrições sejam criadas rapidamente e sua execução não seja atrasada por esforços empreendidos em uma descrição muitas vezes desnecessariamente complexa. Os estágios de um fluxo de trabalho são descritos por elementos chamados processadores em um documento Scufl. Ligações de dados descrevem as dependências de dados entre estágios, que denotam transferências de dados realizadas em tempo de execução. Não é possível aplicar transformações, como consultas XPath, aos dados de saída antes de serem transferidos de um estágio para outro. É possível especificar iterações em estágios e coordenação entre estágios. A especificação de coordenação significa que a execução de um estágio não pode ser iniciada antes do término de outro, mesmo que não haja dependências de dados entre os mesmos.

WS-CDL [W3C, 2005] foi proposta para descrever processos de coreografia que envolvem um conjunto de serviços. Permite descrever as interações entre os diversos serviços participantes de um fluxo de trabalho de um ponto de vista global. Seu principal objetivo é descrever detalhadamente as interações entre serviços que pertencem a diferentes domínios administrativos a fim de que os mesmos sejam desenvolvidos de maneira independente e, ainda assim, sejam capazes de colaborar para a realização de uma mesma tarefa. A linguagem possui suporte a atividades semelhantes às presentes na especificação do WS-BPEL, como: sequência de operações, operações paralelas, escolha dentre um conjunto de operações e execução repetitiva.

## 3.3 Sistemas de gerenciamento de fluxos de trabalho científicos

Nesta seção, são descritas algumas ferramentas existentes para gerenciamento de fluxos de trabalho e são destacadas suas similaridades e diferenças quando comparadas ao caOS. Existem vários sistemas de gerenciamento de fluxos de trabalho, mas a discussão realizada aqui será restrita aos seguintes sistemas: Triana [Taylor et al., 2003], Pegasus [Deelman et al., 2005], Anthill [Tavares et al., 2007] e Taverna [Oinn et al., 2004b].

Triana [Taylor et al., 2003] é um sistema de gerenciamento de fluxos de trabalho que permite a construção, execução e monitoramento de um fluxo de trabalho. Fluxos de trabalho podem ser construídos e executados através de uma interface gráfica. A execução de um fluxo de trabalho usando Triana não é restrita ao sistema de execução desenvolvido para Triana, mas pode ser feito utilizando-se qualquer sistema de execução, desde que este possa ser acessado através de um Triana Controlling Service, um serviço com interface definida e que atua como intermediário na execução de tarefas necessárias para execução de um fluxo de trabalho. Triana executa um fluxo de trabalho por orquestração e, por consequência, todas as transferências de dados são realizados por uma entidade centralizadora.

Pegasus [Deelman et al., 2005] é um sistema de gerenciamento de fluxos de trabalho que realiza o mapeamento de um *fluxo de trabalho abstrato* nos recursos de uma grade para criar um *fluxo de trabalho concreto*. Um *fluxo de trabalho abstrato* é um conjunto de tarefas interconectadas por canais de comunicação que descrevem um fluxo de trabalho sem mencionar os recursos computacionais necessários para sua execução. O mapeamento de tal conjunto no conjunto dos nós de processamento existentes numa grade constitui o *fluxo de trabalho concreto*. O sistema não restringe a execução a um tipo de estágio apenas, conforme mostrado em [Mandal et al., 2007]. Pegasus também implementa mecanismos de recuperação de falhas e pode armazenar dados produzidos durante a execução de um fluxo a fim de que sejam utilizados em execuções posteriores. Isso ocorre quando dados utilizados em certo ponto da execução de um fluxo de trabalho foram produzidos previamente por outro fluxo e o custo de uma segunda produção dos dados é maior que o custo de recuperar dados previamente produzidos.

No trabalho descrito em [Tavares et al., 2007], é apresentado um sistema no qual Anthill [Ferreira et al., 2005], com auxílio de Mobius [Oster e Saltz, 2004], é estendido e torna-se capaz de: criar fluxos de trabalho a partir de uma descrição em formato XML, executar fluxos de trabalho com tolerância a falhas e prover armazenamento distribuído para as entradas, resultados intermediários e saídas de uma execução. No Anthill, um estágio do fluxo de trabalho corresponde a um executável que utiliza troca de mensagens

conforme estabelecido pela MPI [Gropp et al., 1996], logo ele não é capaz de executar fluxos de trabalho formados por serviços Web como os WFMSs descritos anteriormente. Além disso, Anthill foi criado para dar suporte a fluxos de trabalho intensivos em CPU num ambiente de aglomerado de computadores. O modelo filtro-fluxo adotado pelo Anthill possui algumas características singulares que não são encontradas nos demais sistemas, como cópias transparentes de estágios e fluxo rotulado, o que dá ao Anthill a capacidade de rotear mensagens entre cópias transparentes baseando-se no conteúdo das mensagens.

Taverna [Oinn et al., 2004b] é um sistema que foi criado com o objetivo de dar suporte a fluxos de trabalho da área de Biomedicina, embora não se restrinja a estes. Seu componente principal é uma interface gráfica através da qual um fluxo de trabalho pode ser: desenvolvido com o auxílio de diagramas, executado e monitorado. Taverna utiliza não só serviços Web como componentes para criação de um fluxo de trabalho científico, mas também é capaz de executar fluxos de trabalho constituídos por *scripts* Beanshell e RShell, além de um conjunto de serviços analíticos e de acesso a banco de dados disponibilizados pela comunidade de usuários. Wei Tan *et al* [Tan et al., 2008] desenvolveram uma extensão que acrescenta a capacidade de descobrir serviços caGrid e utilizá-los para construir um fluxo de trabalho, fazendo com que Taverna seja capaz de executar fluxos de trabalho constituídos por serviços caGrid. Taverna é capaz de invocar de forma segura uma operação de um serviço Web, mas, para tal, requer que as credenciais utilizadas para uma execução de um fluxo de trabalho sejam, sem auxílio do sistema, enviadas às máquinas em que cada serviço seguro estão hospedados.

O caOS é um WFMS que utiliza coreografia para executar fluxos de trabalho científicos. Como ele evita realizar transferências de dados duplicadas como um sistema que utiliza orquestração pode realizar, também evita as perdas de desempenho correspondentes. O sistema implementa um mecanismo para invocar operações de serviços Web de forma segura, utilizando credenciais delegadas pelo usuário ao caOS através do serviço de delegação de credenciais do caGrid [Langella, 2007]. O caOS é capaz de incorporar a um fluxo de trabalho qualquer serviço Web cuja interface seja descrita por um documento WSDL [W3C, 2007b], pois baseia-se apenas no SOAP, que é um padrão aberto, para invocar operações de serviços Web. Não são implementados mecanismos para recuperação de falhas e reutilização de dados, mecanismos esses que estão presentes em outros WFMSs. No entanto, o caOS foi desenvolvido de forma que novos componentes e mecanismos podem ser acrescentados com facilidade à sua arquitetura modular.

## 3.4 Sumário

Neste capítulo, foram descritos vários trabalhos que estão relacionados ou se assemelham de alguma forma ao trabalho descrito nesta dissertação. As grades computacionais consideradas mais relevantes, algumas linguagens para descrição de fluxos de trabalho científicos e os mais importantes sistemas de gerenciamento de fluxos de trabalho foram descritos.

A seguir, a abordagem escolhida para o problema de criar e executar fluxos de trabalho no caGrid com facilidade e eficiência, que consiste na criação de um sistema de gerenciamento de fluxos de trabalho científicos, é descrita em detalhes no próximo capítulo.

# Capítulo 4

# Coreografia de fluxos de trabalho para computação em grades

Este capítulo discorre sobre a abordagem utilizada para o problema de criar e executar fluxos de trabalho no caGrid com facilidade. Tal abordagem consiste na implementação de um sistema, o *caGrid Choreography System* (caOS), que realiza todas as tarefas necessárias para criar e executar um fluxo de trabalho a partir de uma descrição do mesmo. O caOS possui uma arquitetura com 2 componentes, implementados através de 2 serviços Web: (i) o serviço de suporte à execução (SE), que é responsável pela criação e execução dos elementos de um fluxo de trabalho, e (ii) o serviço de gerenciamento de fluxos de trabalho (SG), que é responsável por interpretar uma descrição do fluxo de trabalho e criar o mesmo. A criação e execução de um fluxo de trabalho é feita pelo SG através de chamadas ao SE, mas apenas o SG é capaz de controlar um fluxo de trabalho como um todo. O desenvolvimento do caOS foi realizado dentro do ambiente de incubação de projetos do caGrid, o que exigiu que tal desenvolvimento fosse feito em conformidade com as diretrizes estabelecidas para integração futura ao projeto principal. Entre tais diretrizes destacam-se: a adoção de padrões estabelecidos, a realização de testes automatizados no sistema e a disponibilzação pública do código-fonte.

Nas seções seguintes, são descritos a arquitetura do caOS, os mecanismos implementados para permitir a invocação de serviços Web de forma segura e a interface do caOS com o usuário, que inclui o formato da descrição de fluxos de trabalho utilizada neste trabalho. A Seção 4.1 descreve o serviço de suporte à execução em detalhes. O serviço de gerenciamento de fluxos de trabalho científicos é descrito na Seção 4.2. Em seguida, a interface do caOS com o usuário é descrita na Seção 4.3. Os mecanismos implementados para a utilização de serviços seguros em fluxos de trabalho, dos quais participam tanto o SE quanto o SG, são descritos na Seção 4.4. A Seção 4.5 sumariza

o conteúdo deste capítulo.

## 4.1 Serviço de suporte à execução

O serviço de suporte à execução (SE) centraliza a criação e configuração dos estágios de um fluxo de trabalho e controla a execução dos mesmos. Foi implementado como um serviço Web, escrito em Java com auxílio do Introduce [Hastings et al., 2007], capaz de criar 2 tipos de contexto: instâncias de estágios e instâncias de fluxo de trabalho local. As instâncias de estágios, descritas na Seção 4.1.1, são responsáveis por invocar um estágio de um fluxo de trabalho e transferir dados a partir deste. As mesmas estão sempre associadas a uma instância de fluxo de trabalho local. As instâncias de fluxo de trabalho local, descritas na Seção 4.1.2, representam o conjunto formado pelos contextos que estão associados aos componentes de um fluxo de trabalho localizados no mesmo servidor Web (note que todos os componentes do caOS são serviços Web). As instâncias de fluxo de trabalho local são também responsáveis por recuperar as credenciais delegadas pelo usuário ao caOS, armazená-las e entregá-las às instâncias de estágio que precisam das mesmas para invocar uma operação de um serviço seguro. Os mecanismos que utilizam credenciais para invocar operações de serviços seguros são detalhados na Seção 4.4.

Para instanciar um fluxo de trabalho utilizando o SE, deve-se criar uma instância de fluxo de trabalho local (IFTL) em primeiro lugar. Através da IFTL criada, podem ser criadas as instâncias de seus estágios, conforme ilustrado na Figura 4.1. Instâncias de estágios não podem ser criadas diretamente pelo SE, pois é necessário garantir que todas as instâncias de estágio possuam uma associação com uma instância de fluxo de trabalho local. Assim, a interface do SE possui apenas um método, responsável pela criação de uma instância de fluxo de trabalho local. Para executar os estágios, é necessário prover às instâncias de estágios as entradas necessárias à execução e iniciar a execução de cada estágio invocando os métodos correspondentes nas instâncias de estágio.

### 4.1.1 Instâncias de estágios

Uma instância de estágio é um contexto do SE que representa um estágio do fluxo de trabalho. Uma vez associada a uma operação de um serviço Web, uma instância de estágio atua como intermediária em todas as ações que o caOS pode realizar em relação àquela operação. A seguir, serão descritas as características presentes numa instância de estágio, os passos necessários à criação da mesma e à execução da operação associada no contexto de um fluxo de trabalho.

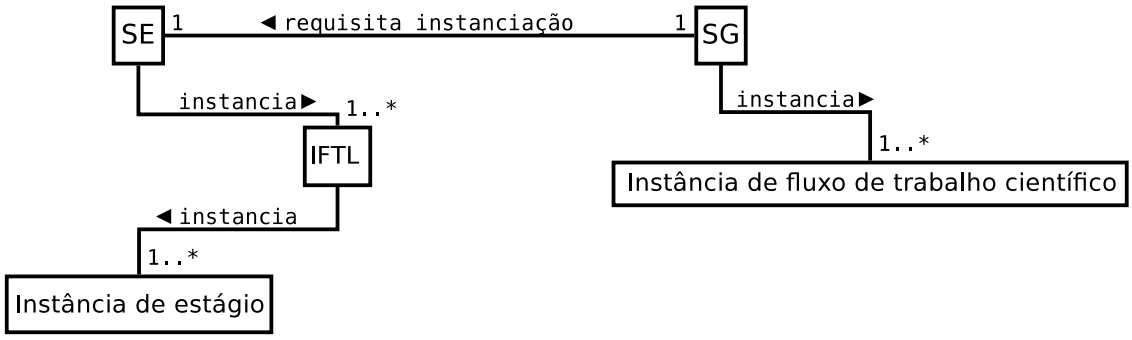


**Figura 4.1.** Relacionamentos entre o serviço de suporte à execução e seus contextos

Uma característica fundamental para que a utilização de um sistema de gerenciamento de fluxos de trabalho seja simples é a capacidade de invocar serviços Web sem os clientes para acesso aos serviços, criados durante o desenvolvimento destes. No caOS, uma instância de estágio pode invocar qualquer serviço Web que seja capaz de processar mensagens do tipo SOAP [W3C, 2007a], que é um padrão aberto amplamente empregado. Uma instância de estágio é capaz de construir uma mensagem SOAP e utilizá-la para invocar uma operação específica de um serviço Web. A instância de estágio é associada sempre a uma única operação em particular e, por isso, deve ser configurada com as seguintes informações da operação que se deseja invocar: URL do serviço Web, nome da operação, tipo de retorno e uma lista de pares $\langle nome,\ tipo \rangle$ correspondente à lista de parâmetros. Estas informações, além dos valores dos parâmetros de entrada, são necessárias para a criação da mensagem de invocação. Tal capacidade faz com que o desenvolvimento de um fluxo de trabalho demande um esforço menor, mudanças na implementação de um serviço acarretem poucas mudanças para que o mesmo seja executado pelo caOS, além da já mencionada compatibilidade com uma quantidade muito grande de serviços. Uma instância de estágio é capaz de invocar tanto um serviço que esteja no servidor Web local quanto em um servidor remoto, embora a invocação local tenha o benefício de evitar os custos, muitas vezes altos, de uma chamada remota feita na grade computacional.

A existência de serviços cuja execução é muito longa torna a capacidade de executar serviços de forma assíncrona uma característica também importante. No caOS, uma instância de estágio possui a capacidade de executar um estágio cuja execução é muito longa. O início de uma execução é realizado de maneira assíncrona pela operação *iniciaExecução*. Durante sua execução, a operação *iniciaExecução* cria um fluxo de execução além do principal e retorna, mantendo o fluxo de execução secundário ainda ativo. Após concluir sua execução, tal fluxo secundário envia uma notificação, que obedece ao padrão *WS-Notification* [OASIS, 2006a], a quem invocou a operação. Assim,

a notificação é interpretada por quem a recebe como um sinal de que a chamada assíncrona foi concluída. É importante ressaltar que a invocação de uma operação de um serviço Web em si não é necessariamente assíncrona e operações que não foram implementadas para serem executadas de forma assíncrona utilizam um tempo limite de espera para saber quando uma invocação falha. Para amenizar este problema, uma instância de estágio configura o tempo limite de espera da chamada com o maior valor possível. Esta configuração permite que operações permaneçam em execução por até aproximadamente 24 dias.

A instância de estágio também é o principal responsável pela capacidade do caOS de utilizar coreografia para executar um fluxo de trabalho. Ela pode ser configurada para transferir os dados de saída do estágio pelo qual é responsável para outros estágios. Mais precisamente, ela pode entregar dados para outras instâncias de estágio. Como as instâncias de estágio realizam as transferências de dados independentemente, não há necessidade de um elemento central atuante em todas as transferências entre instâncias de estágio e a coreografia é empregada ao invés de orquestração. O dado de saída pode ser transformado antes de ser transferido, através da aplicação de uma consulta XPath [W3C, 2007c] sobre ele a fim de obter uma parte específica. A configuração das transferências de dados que uma instância de estágio deve realizar consiste numa lista de pares formados por uma consulta XPath [W3C, 2007c] a ser aplicada ao dado de saída e o endereço da instância de estágio à qual o dado resultante deve ser enviado.

Para criar uma instância de estágio, é necessário requerer a criação da mesma à instância de fluxo de trabalho local à qual deseja-se associar o novo contexto. A operação *criaInstânciaDeEstágio* é a operação a ser utilizada para tal e deve receber a descrição da instância a ser criada conforme descrito anteriormente. Uma vez criada, uma instância de estágio é configurada tanto com sua lista de parâmetros, através da operação *configuraEntradas*, quanto com a descrição das transferências de dados a realizar, feita através da operação *configuraTransporteDaSaída*. Neste ponto, a configuração da operação a invocar está completa. Para executar o estágio associado, os valores dos parâmetros de entrada devem ser fornecidos através da operação *assinalaEntrada* e a execução deve ser iniciada através da operação *iniciaExecução*. Os valores dos parâmetros de entrada devem ser fornecidos no formato XML [W3C, 2008], pois esse é o formato mais apropriado para inclusão de um objeto na requisição SOAP [W3C, 2007a] utilizada para iniciar a execução do estágio.

Para possibilitar o monitoramento da execução de um estágio, foi criado um conjunto de estados nos quais uma instância de estágio pode estar. Assim, a cada operação executada por uma instância de estágio, uma mudança de estado pode ocorrer. Um diagrama com os estados criados e as operações que geram transições entre estes é mos-

trado na Figura 4.2. É possível monitorar as mudanças de estado de uma instância de estágio através de notificações, o que possibilita, por exemplo, executar ações de acordo com o estado atual de uma instância de estágio. Uma aplicação pode cadastrar-se para receber notificações de uma instância de estágio utilizando a operação *cadastraMonitoramento*.

Uma instância de estágio também pode executar um estágio por várias iterações. Isso é feito sempre que se verifica que foram fornecidos mais dados de entrada como valor de um parâmetro do que o necessário para uma execução, em outras palavras, é fornecido um arranjo em lugar de um único objeto. Quando isso ocorre, o estágio correspondente é executado uma vez para cada elemento pertencente ao arranjo fornecido.

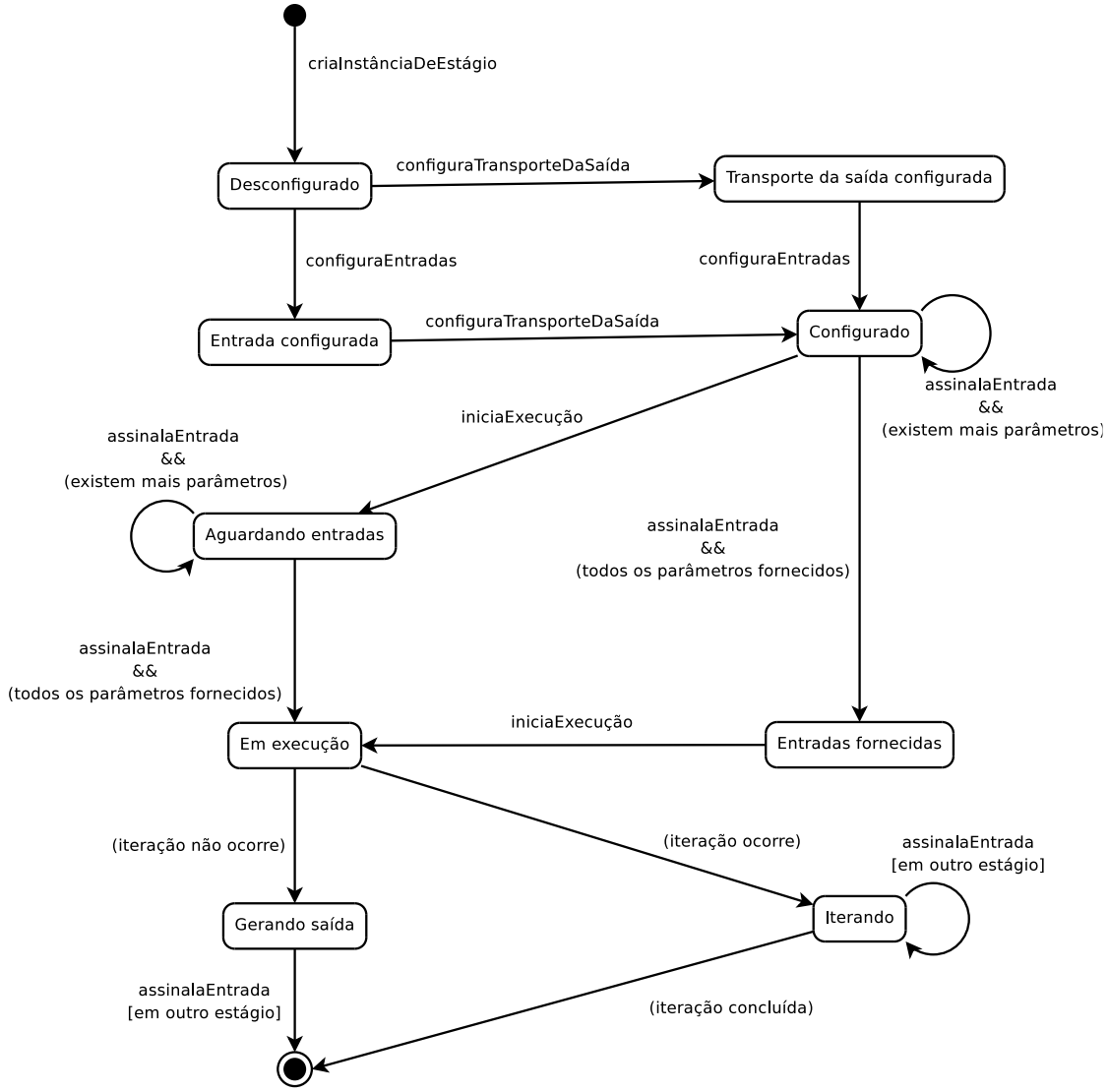


**Figura 4.2.** Diagrama dos estados de uma instância de estágio

### 4.1.2 Instâncias de fluxo de trabalho local

Uma instância de fluxo de trabalho local (IFTL) é um contexto do SE que representa um fluxo de trabalho local, conforme definido na Seção 2.1. Tal contexto é responsável pela criação de instâncias de estágio e atua também controlando o uso de credenciais dentro do SE.

A criação de instâncias de estágio é realizada através do método *criaInstânciaDeEstágio* constante da interface de uma IFTL e que tem como parâmetro a descrição citada na Seção 4.1.1. A partir do contexto recém-criado, é possível configurar e executar um estágio do fluxo de trabalho local. É importante mencionar que, apesar da definição prever apenas um fluxo de trabalho local por máquina hospedeira, o caOS não impõe restrições quanto ao número de instâncias de tal contexto criadas numa mesma máquina, ou seja, um fluxo de trabalho pode ter mais de uma IFTL por máquina. Um motivo que pode levar à criação de várias IFTLs por máquina é o balanceamento da carga correspondente ao controle das instâncias de estágio.

Uma IFTL é responsável por recuperar e armazenar credenciais, além de fornecê-las às instâncias de estágio que as utilizam para invocar serviços de forma segura. Através do método *acrescentaCredencial* é possível associar uma credencial a uma instância de estágio. O parâmetro do método é, na verdade, o endereço de uma credencial delegada ao SE através do *caGrid Credential Delegation Service* (CDS) [Langella, 2007] conforme discutido na Seção 2.1 a fim de que o SE e todos os contextos criados por ele possam identificar-se e agir em nome do dono da credencial. Tal endereço é utilizado, então, pela IFTL para recuperar uma credencial do CDS e associá-la a uma instância de estágio existente. Caso uma credencial seja associada a vários estágios, ela é recuperada do CDS apenas ao fazer a primeira associação e é apenas associada aos demais estágios. Uma vez que uma instância de estágio do fluxo de trabalho local requisite uma credencial para invocar um serviço seguro, a IFTL fornece a credencial associada à instância de estágio. É possível substituir uma associação entre uma credencial e um estágio por uma associação com outra credencial. Isso pode ocorrer, por exemplo, quando uma credencial delegada está próxima de expirar. Feita uma substituição como a citada, a nova credencial será fornecida em resposta às subsequentes requisições por parte de instâncias de estágio.

O monitoramento do estado de cada instância de estágio de um fluxo de trabalho local também é responsabilidade de uma IFTL. Tal tarefa é realizada com auxílio dos tópicos de notificação disponibilizados por uma instância de estágio, através dos quais é possível a uma IFTL cadastrar-se, usando a operação *cadastraMonitoramento*, para ser informada de mudanças no estado de um estágio. Um papel relacionado é o de determinar o estado de um fluxo de trabalho local e assinalar tal valor a um

tópico de notificação disponibilizado pelo IFTL, o que permite o envio de notificações a entidades que cadastrem-se para serem informadas sobre mudanças de estado do IFTL. A determinação do estado atual de um fluxo de trabalho local é feita segundo o diagrama da Figura 4.3. Assume-se que uma IFTL executa determinada transição de estado somente após todas as instâncias de estágio associadas executarem tal transição. A fim de diminuir o número de notificações, uma notificação de mudança de estado só é emitida quando o estado do fluxo de trabalho local é alterado. No entanto, os estados de todos os estágios são enviados junto com a notificação a fim de que seja possível determinar, por exemplo, o estágio exato em que ocorreu uma falha.

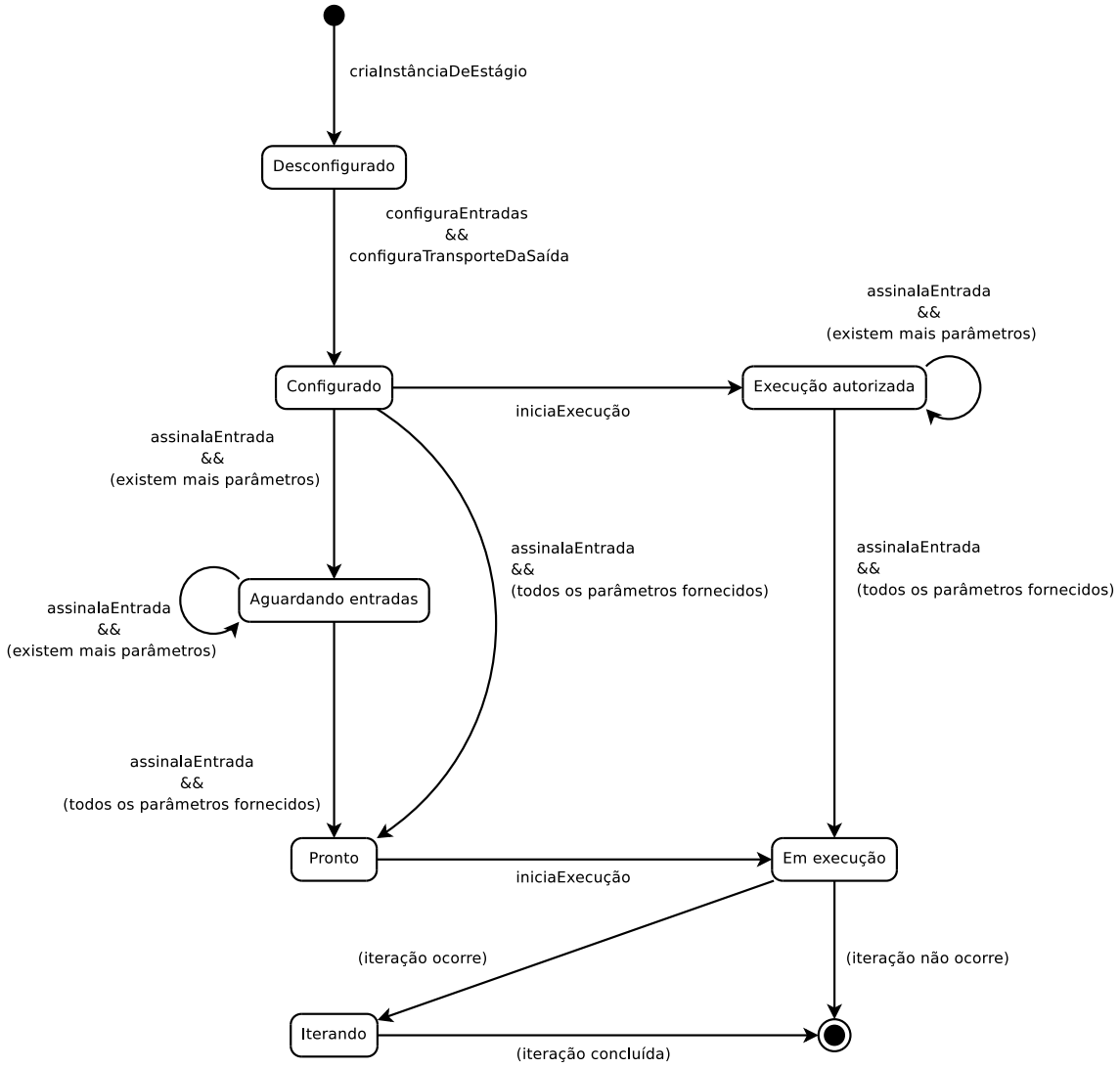


**Figura 4.3.** Diagrama dos estados de uma instância de fluxo de trabalho local

## 4.2 Serviço de gerenciamento de fluxos de trabalho

O serviço de gerenciamento de fluxos de trabalho (SG) é o componente do caOS responsável pela criação de um fluxo de trabalho. Foi implementado como um serviço Web capaz de criar um contexto chamado de instância de fluxo de trabalho, que representa um fluxo de trabalho completo. Sua interface possui apenas um método, *criaInstânciaDeFluxoDeTrabalho*, cuja função é criar uma instância de fluxo de trabalho.

A criação de um fluxo de trabalho se dá a partir de uma descrição de seus componentes, suas entradas e saídas. Com tais informações o SG é capaz de criar uma instância de fluxo de trabalho, que invoca o SE para criar os contextos correspondentes a cada componente do fluxo de trabalho. O contexto criado é responsável por executar e monitorar o fluxo de trabalho associado e está descrito na subseção seguinte.

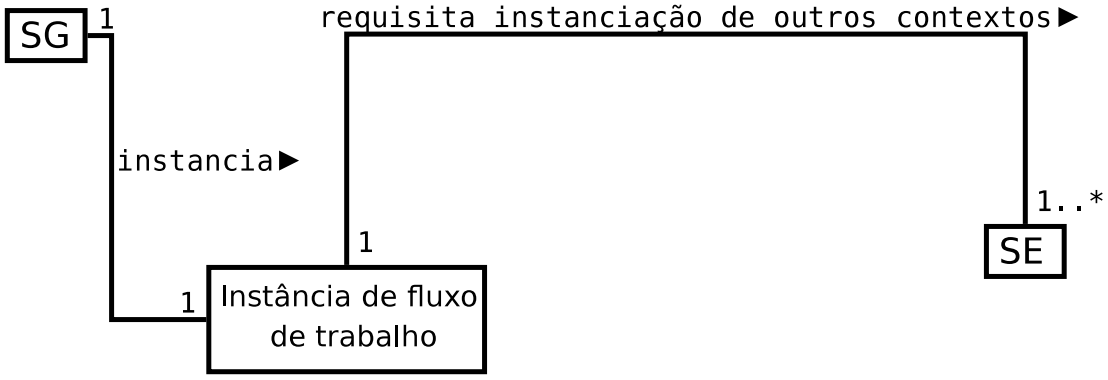


**Figura 4.4.** Relacionamentos entre o SG, seus contextos e o SE

### 4.2.1 Instância de fluxo de trabalho

Uma instância de fluxo de trabalho, que é um contexto criado pelo SG, representa um fluxo de trabalho no caOS. Através dele é possível executar e monitorar um fluxo de trabalho.

Durante sua criação, uma instância de fluxo de trabalho recebe a descrição de seus componentes: os fluxos de trabalho locais e os estágios pertencentes a estes. Através do SE, são criadas as instâncias de fluxo de trabalho locais e as instâncias de estágio correspondentes aos componentes do fluxo de trabalho. As instâncias de estágio são configuradas com as descrições das operações de serviço Web correspondentes antes de qualquer delas ser configurada com o transporte dos dados de saída, pois é necessário que o endereço de todas as instâncias de estágio seja conhecido para associar um endereço a cada transferência de dados a ser executada. As instâncias de estágio cujos dados de saída correspondem aos dados de saída do fluxo de trabalho são configuradas

para enviar tais dados para a instância de fluxo de trabalho. Para realizar o recebimento dos dados de saída do fluxo de trabalho, uma instância de fluxo de trabalho possui uma operação, chamada *assinalaEntrada*, idêntica à operação utilizada para enviar dados de entrada a uma instância de estágio. Ela foi importada da instância de estágio, usando o Introduce, e, por isso, uma instância de estágio não necessita executar uma operação diferente para enviar dados cujo destino é uma instância de fluxo de trabalho. Uma instância de estágio, na verdade, não é capaz de diferenciar os dois tipos de destinatário de suas transferências de dados.

Uma instância de fluxo de trabalho é capaz de monitorar as mudanças de estado dos fluxos de trabalho locais, cadastrando-se para receber notificações sobre mudanças no tópico correspondente das instâncias de fluxo de trabalho locais. A partir destes, o estado do fluxo de trabalho é determinado segundo o mesmo diagrama de estados das instâncias de fluxo de trabalho local mostrado na Figura 4.3. O estado de uma instância de fluxo de trabalho também é assinalado a um tópico, ao qual um programa externo ao caOS pode cadastrar-se, usando a operação *cadastraMonitoramento*, para ser notificado sobre mudanças de estado no fluxo de trabalho.

## 4.3 Interface com o usuário

A fim de facilitar a utilização do caOS, foi criada uma biblioteca de classes Java que auxiliam na construção da descrição de um fluxo de trabalho científico utilizada pelo sistema. Tal biblioteca permite que um arquivo XML [W3C, 2008] seja interpretado e, a partir deste, um objeto do tipo utilizado pelo caOS para descrever um fluxo de trabalho seja construído. De posse do objeto construído, é possível utilizar o SG para criar uma instância de fluxo de trabalho (operação *criaInstânciaDeFluxoDeTrabalho*), executar o fluxo de trabalho correspondente (operação *iniciaExecução*) e recuperar os dados de saída após a execução (operação *recuperaDadosDeSaída*). Na seção seguinte, são descritos as informações constantes da descrição de um fluxo de trabalho e o formato do arquivo XML aceito pela interface.

### 4.3.1 Descrição de um fluxo de trabalho científico

Nesta seção, o formato da descrição de um fluxo de trabalho utilizada neste trabalho é descrito. Tal descrição será feita em níveis crescentes de detalhamento, começando com um fluxo de trabalho completo e prosseguindo até descrever as características de cada estágio. O Apêndice B contém a definição da descrição utilizada e de cada um de seus elementos na forma de esquemas.

A descrição de um fluxo de trabalho científico foi dividida, inicialmente, em 3 partes: a descrição dos dados de entrada necessários para sua execução, a descrição dos dados de saída produzidos e a descrição dos fluxos de trabalho locais. A descrição dos dados de entrada consiste, para cada parâmetro do fluxo de trabalho, em: o valor do parâmetro em forma de texto XML [W3C, 2008], um identificador numérico associado ao estágio que deve receber o dado e o índice do parâmetro ao qual o valor recebido deve ser assinalado. Os dados de saída são descritos, para cada dado, por: um identificador associado ao estágio do qual será obtido o valor do dado, uma consulta XPath [W3C, 2007c] a ser aplicada sobre o dado de saída do estágio para que o valor do dado de saída do fluxo de trabalho seja obtido, o tipo do dado e um identificador numérico utilizado para diferenciá-lo dos demais dados de saída do fluxo de trabalho. A descrição dos fluxos de trabalho locais será definida a seguir.

A descrição de cada fluxo de trabalho local possui 2 partes: o endereço (URL) do SE utilizado para criar a instância de fluxo de trabalho local e a descrição de cada estágio pertencente ao mesmo. A descrição de um estágio apresenta: a assinatura da operação de serviço Web associada, as transferências que o estágio deve realizar utilizando os dados resultantes de sua execução, os requisitos de segurança para a execução da operação e um identificador numérico que diferencia o estágio de todos os demais estágios pertencentes ao fluxo de trabalho. A assinatura de um estágio é descrita por: o nome da operação, o tipo de retorno e uma lista com os nomes e tipos de cada parâmetro de entrada. Uma transferência realizada por um estágio é descrita utilizando: o identificador numérico correspondente ao estágio que é o destino da transferência, uma consulta XPath [W3C, 2007c] que deve ser aplicada aos dados de saída para obter o dado utilizado pelo destinatário da transferência, o tipo esperado do dado após transformado pela consulta XPath e um índice numérico que indica a qual parâmetro do estágio destinatário deve ser assinalado o valor do dado transferido.

A descrição dos requisitos de segurança necessários à execução de um estágio merece destaque. É possível especificar operações que obedecem aos seguintes esquemas de segurança: mensagem segura, conversação segura e transporte seguro (TLS). Tais esquemas são definidos, junto com os esquemas de autenticação, pelo componente *Grid Security Infrastructure* (GSI) [Foster et al., 1998] pertencente ao Globus [Foster, 2005] e serão descritos na Seção 4.4. Ainda é possível especificar o tipo de proteção do canal de comunicação como privacidade (comunicação criptografada) e/ou integridade (comunicação assinada). A descrição do esquema de segurança de um estágio se dá pela especificação de um, e apenas um, dos 3 esquemas de segurança citados acima. A descrição do esquema de mensagem segura contém o nível de proteção do canal de comunicação e o esquema de autenticação exigido: autenticação no serviço ou autenti-

cação utilizando o serviço de delegação de credenciais do caGrid (CDS) [Langella, 2007]. A descrição do esquema de conversação segura consiste no nível de proteção do canal e no esquema de autenticação, que pode assumir os valores: autenticação no serviço, autenticação utilizando o CDS ou autenticação por delegação nativa. O esquema de transporte seguro é descrito, além do nível de proteção do canal, por: autenticação no serviço ou autenticação utilizando o CDS. Note que a forma como o esquema de transporte seguro aplica proteção ao canal de comunicação difere do esquema de mensagem segura, conforme é descrito na Seção 4.4.

Através de uma descrição de um fluxo de trabalho científico que obedece ao formato descrito acima, é possível criar e executar fluxos de trabalho utilizando o caOS. Todos os componentes de um fluxo de trabalho podem ser descritos, assim como a interação entre os mesmos. Além disso, esquemas de segurança utilizados por operações componentes de um fluxo de trabalho podem ser descritas, exigindo-se apenas que estes estejam em conformidade com os esquemas definidos pelo GSI.

## 4.4 Mecanismos para invocações seguras

Nesta seção, são descritos os aspectos relacionados à segurança presentes no caOS. Em primeiro lugar, os esquemas de segurança aceitos pelo caOS para invocação de estágios são descritos. Em seguida, o processo pelo qual o caOS torna possível invocar operações de forma segura é descrito, destacando-se o papel desempenhado por cada componente do caOS.

Os esquemas de segurança que um estágio pode exigir para sua execução são: mensagem segura, conversação segura e transporte seguro. Tais mecanismos correspondem àqueles definidos pelo GSI [Foster et al., 1998] e todos exigem credenciais para que seja aplicada proteção aos canais de comunicação. O esquema de mensagem segura atua apenas no conteúdo de uma mensagem SOAP [W3C, 2007a], deixando o cabeçalho intacto. Durante uma invocação, tal conteúdo é protegido conforme especificado na descrição do esquema - criptografado no caso de privacidade e assinado no caso de integridade - e o restante da mensagem SOAP não sofre alteração. O esquema de conversação segura atua de uma forma muito semelhante ao esquema anterior, mas é criado um contexto seguro no serviço a ser invocado que pode ser utilizado durante várias invocações. O esquema de transporte seguro consiste em utilizar toda uma mensagem SOAP ao aplicar a proteção do canal, ou seja, todas as partes de uma mensagem SOAP são criptografadas ou assinadas durante a invocação de uma operação de um serviço Web.

No caOS, várias entidades participam do processo de configuração e execução das

invocações seguras de operações de serviços Web. É participante do processo o usuário ou programa que deseja executar um fluxo de trabalho. Todos os contextos que representam componentes de um fluxo de trabalho no caOS também atuam em tal processo. Além destes, o *caGrid Credential Delegation Service* (CDS) [Langella, 2007] atua de forma fundamental em todo o processo, realizando delegação de credenciais conforme descrito na Seção 2.1. Os mecanismos de segurança do caOS são ilustrados na Figura 4.5. É importante ressaltar que tais mecanismos só são executados quando parte dos estágios de um fluxo de trabalho precisa ser acessada de forma segura. No caso em que não há tais estágios, não há necessidade de executar os mecanismos descritos no restante desta seção e, por isso, os mesmos não são executados.

O CDS é utilizado tanto pelo usuário como pelo SG e pelas instâncias de fluxo de trabalho local criadas pelo SE. A função do CDS nos mecanismos de segurança do caOS é viabilizar a delegação e a recuperação de credenciais. Através do CDS, é possível a um elemento $A$ delegar seus direitos a outro elemento $B$, conforme descrito na Seção 2.1 e ilustrado na Figura 2.2. O elemento $B$ pode, então, invocar serviços em nome de $A$. Para restringir as ações que $B$ pode tomar em nome de $A$, uma credencial delegada expira após um período de tempo escolhido por $A$. Além disso, $B$ pode delegar a mesma credencial a um terceiro, mas $A$ pode restringir o número de vezes que a credencial pode ser delegada a partir de $B$, podendo, inclusive, não permitir tal ação.

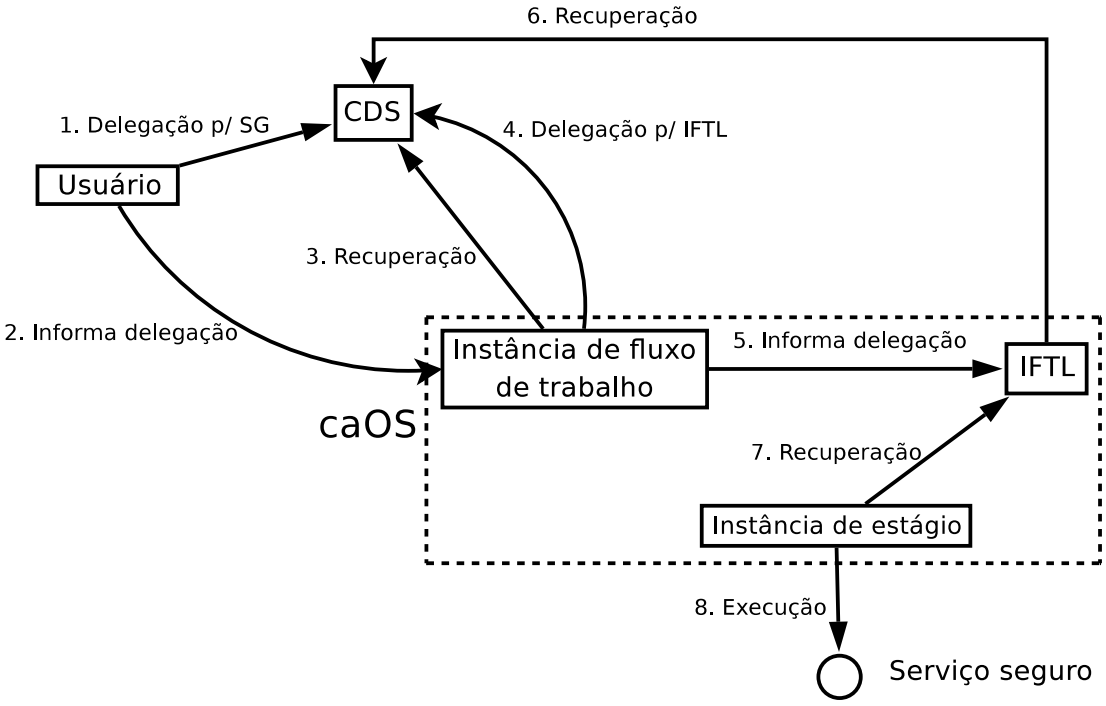


**Figura 4.5.** Interações existentes nos mecanismos de segurança do caOS

O usuário (ou programa) que executa um fluxo de trabalho através do caOS deve possuir credenciais que o permitam executar cada estágio do fluxo de trabalho que exija acesso seguro. Tais credenciais devem, então, ser delegadas ao SG através do CDS e incluídas na descrição do fluxo de trabalho que é enviada ao SG. As credenciais delegadas precisam ser configuradas de tal forma que o SG seja capaz de delegá-las a um terceiro sujeito, do contrário apenas o SG poderá utilizar as credenciais e as instâncias de estágios não poderão fazê-lo. As credenciais delegadas poderão, após serem recuperadas pelo SG, ser utilizadas para agir em nome do usuário. Note que o caOS permite utilizar credenciais diferentes para executar estágios diferentes e também permite que credenciais sejam substituídas, o que pode ocorrer quando estão próximas de expirar.

A instância de fluxo de trabalho criada pelo SG participa dos mecanismos de segurança executados pelo caOS interagindo com outros 3 participantes do processo: o usuário, o CDS e as instâncias de fluxo de trabalho local criadas pelo SE. Uma vez que o usuário delega suas credenciais à instância de fluxo de trabalho utilizando o CDS e entrega uma descrição do fluxo de trabalho que deseja criar à mesma, esta é capaz de obter as credenciais delegadas e agir em nome do usuário. Como espera-se que os fluxos de trabalho locais e as instâncias de estágios sejam criadas em servidores Web diferentes daquele no qual a instância de fluxo de trabalho está, as credenciais recebidas pela mesma precisam ser novamente delegadas, dessa vez às instâncias de fluxo de trabalho local (IFTLs, contextos criados pelo SE) responsáveis pelos estágios que utilizam as credenciais. Após realizar as delegações necessárias, a instância de fluxo de trabalho envia às IFTLs os endereços através dos quais estas poderão obter as credenciais delegadas. Tais credenciais, embora delegadas pela instância de fluxo de trabalho, são associadas ao usuário e não ao SG. Isso permite que uma IFTL atue em nome do usuário ao utilizar as credenciais recebidas.

As instâncias de fluxo de trabalho locais (IFTLs) também interagem com o CDS, além de com as instâncias de estágio pelas quais é responsável. Uma IFTL que é responsável por instâncias de estágio que utilizam credenciais obtém as credenciais delegadas pela instâncias de fluxo de trabalho através do CDS. Tais credenciais são armazenadas pela IFTL numa estrutura de dados que associa cada estágio que utiliza uma credencial à credencial correspondente. Para acessar uma credencial armazenada, é necessário fornecer o identificador da instância de estágio associada à credencial. Isso evita acessos de terceiros às credenciais armazenadas, pois o identificador das instâncias de estágios participantes de um fluxo de trabalho científico é gerado no momento de sua criação e não é acessível de nenhuma forma fora do caOS. Caso um identificador de instância de estágio seja forjado, não será possível alterar as associações existentes entre

credenciais e instâncias de estágios, mantendo, assim, a autenticidade das credenciais e das associações. É importante notar também que as instâncias de estágio, não as IFTLs, são capazes de executar estágios e, por isso, precisam ter permissão de usar as credenciais do usuário para agir em seu nome. No entanto, uma IFTL faz com que as instâncias de estágio possuam identidades iguais à sua no momento de suas criações. Assim, todas as credenciais delegadas a uma IFTL podem também ser usadas pelas instâncias de estágio criadas pela mesma IFTL sem nenhum processo de delegação. Quando uma instância de estágio requisita uma credencial, a IFTL entrega a credencial associada ou a informa que não há associação de credencial alguma com a instância de estágio. Uma IFTL permite, ainda, que credenciais sejam substituídas, o que é importante quando alguma credencial está próxima de expirar.

As instâncias de estágio, contextos criados pelo SE, são os destinatários finais das credenciais delegadas pelo usuário. Obviamente, este somente é o caso para as instâncias de estágio que realizam invocações seguras a operações de serviços Web. As instâncias de estágio interagem com a IFTL responsável e com o serviço Web que possui a operação que deve ser executada como parte do fluxo de trabalho. Uma instância de estágio interage com uma IFTL durante a configuração de uma execução, requisitando a credencial da qual necessita para configurar uma execução segura. Uma instância de estágio pode executar a operação associada múltiplas vezes - por exemplo, quando ocorre iteração - e uma credencial é requisitada ao IFTL a cada execução a fim de que credenciais que tenham sido substituídas sejam utilizadas no lugar de uma credencial utilizada anteriormente. Uma instância de estágio também interage com o serviço que disponibiliza a operação a executar. Tal interação ocorre através da mensagem SOAP [W3C, 2007a] que requisita a execução segura de uma operação. A mensagem SOAP é configurada com os parâmetros de segurança recebidos durante a criação da instância de estágio e a credencial correspondente é utilizada para criptografar a mensagem, para assiná-la ou é enviada como parte da mensagem a fim de que o serviço conheça a identidade do usuário dono da credencial e permita ou recuse-se a realizar uma execução.

## 4.5 Sumário

Neste capítulo, foi descrita a forma como foi abordado o problema de criar e executar fluxos de trabalho científicos no caGrid com facilidade e eficiência. O caOS, sistema construído para solucionar tal problema, foi descrito na forma de uma arquitetura composta por serviços Web que permitem a criação e execução de fluxos de trabalho a partir de uma descrição XML [W3C, 2008]. Foram destacados os mecanismos que permitem

a execução de estágios de forma segura e o formato adotado para a descrição de um fluxo de trabalho. No próximo Capítulo, o ambiente em que o caOS foi desenvolvido será descrito, abordando tanto aspectos gerenciais quanto operacionais.

# Capítulo 5

# Ambiente de desenvolvimento

Neste capítulo, o ambiente em que o trabalho descrito nesta dissertação foi desenvolvido será descrito. Tal descrição compreende alguns aspectos gerenciais, como o processo adotado pelos coordenadores do projeto para acompanhá-lo, e aspectos operacionais, como as características do repositório utilizado para o código-fonte, as diretrizes para utilização de tal repositório e o processo de teste contínuo do sistema.

O restante deste capítulo está organizado como segue. A Seção 5.1 descreve como o desenvolvimento do projeto foi acompanhado por seus coordenadores. Em seguida, a Seção 5.2 apresenta o repositório de incubação de projetos do caGrid utilizado no desenvolvimento do caOS e as diretrizes para participação no mesmo. Os testes empregados para garantia de qualidade do sistema e o ambiente utilizado para execução dos mesmos é descrito na Seção 5.3.

## 5.1 Acompanhamento do projeto

Esta seção descreve como ocorreu o acompanhamento do desenvolvimento do projeto do caOS por parte de seus coordenadores. O projeto foi desenvolvido pelo autor desta dissertação no *Department of Biomedical Informatics* (BMI) da *Ohio State University* (OSU) e os coordenadores do projeto faziam parte desse mesmo departamento durante todo o desenvolvimento do caOS. É importante destacar que o BMI realiza pesquisa tanto na área de Ciência da Computação como em Biomedicina e, como consequência, existe uma diversidade de projetos e pesquisadores no departamento.

As principais formas de acompanhamento do desenvolvimento do projeto do caOS foram reuniões e revisões de código. Foram realizadas reuniões em 3 formatos diferentes, cada um com objetivos e periodicidades diferentes. O primeiro tipo de reunião era realizado com o intuito de acompanhar aspectos do projeto em detalhes e discutir soluções para pequenos problemas. Tais reuniões ocorriam 2 vezes por semana, com

duração de 20 minutos, e contavam com a participação de um grupo pequeno constituído por entre 6 e 10 pessoas participantes de um número restrito de projetos do BMI que compartilhavam o espaço físico de um único laboratório de computadores. O segundo tipo de reuniões tinha o objetivo de acompanhar o cumprimento das metas de cada projeto relacionado à área de Ciência da Computação. Tais reuniões eram realizadas 1 vez por semana, com duração de 1 hora, e o trabalho realizado na semana correspondente, assim como as próximas etapas dos projetos, eram apresentados brevemente. O terceiro tipo de reunião corresponde a reuniões para planejamento do projeto do caOS e discussão de soluções de problemas mais complexos. Tais reuniões não eram realizadas com periodicidade e duração definidas, mas eram realizadas conforme o avanço do desenvolvimento do projeto e o surgimento de problemas. Além das reuniões, a realização de revisões do código-fonte do caOS foi planejada para ocorrer ao fim do desenvolvimento de cada serviço do caOS. No entanto, não foi possível realizar nenhuma revisão de código durante o desenvolvimento do caOS.

## 5.2 Incubação de projetos do caGrid

O desenvolvimento do caOS foi realizado de forma integrada ao ambiente de incubação de projetos do caGrid [caGrid, 2008] e tal ambiente será descrito nesta seção, assim como as diretrizes associadas à sua utilização.

O principal recurso presente no ambiente de incubação de projetos do caGrid é o repositório de código-fonte. Tal repositório possui mecanismos de controle de versão de arquivos. É possível, também, que um projeto como o caOS satisfaça suas dependências em relação a outros projetos de forma automática, através da criação de *releases* de cada projeto pertencente ao repositório e sua atualização automatizada. Tomando o caOS como exemplo, existe uma dependência em relação à biblioteca de acesso ao CDS [Langella, 2007] para prover as funcionalidades de delegação e recuperação de credenciais e executar os mecanismos de segurança do caOS (veja Seção 4.4 para detalhes). Tal dependência é satisfeita automaticamente no processo de compilação do caOS devido à disponibilização, no repositório de incubação de projetos do caGrid, da versão mais atual da biblioteca de acesso ao CDS.

A utilização do ambiente de incubação de projetos do caGrid é condicionada à algumas diretrizes. Estas têm como objetivo padronizar os projetos a fim de que os mesmos possam ser integrados sem problemas à distribuição principal do caGrid, se for do interesse da comunidade. A principal regra associada à participação na incubadora é o desenvolvimento de projetos com código aberto. Outra recomendação é a separação, no repositório de código-fonte, do código correspondente ao projeto do

código correspondente aos testes. Tal diretriz têm como objetivo permitir a automatização da realização de testes e da criação das *releases* do projeto, conforme descrito na Seção 5.3. Cada parte do repositório do caOS é subdividida numa linha de desenvolvimento principal (tronco) e linhas de desenvolvimento paralelas (galhos). Além do descrito acima, o repositório do caOS possui um painel de controle (*dashboard*) através do qual é possível observar a execução automática de testes e examinar seus resultados.

## 5.3 Ambiente de testes

Esta seção apresenta os testes realizados para garantia de qualidade do caOS e o ambiente em que os mesmos foram realizados. Os testes são realizados utilizando o código-fonte armazenado no repositório de incubação de projetos do caGrid. São realizados testes de unidade, integração e sistema de forma automatizada. A definição de cada tipo de teste pode ser encontrada em [de Pádua Filho, 2009]. Para todos os testes, é utilizado um computador que monitora o repositório de código-fonte em busca de alterações. O mesmo computador também é capaz de executar uma sequência de comandos que recupera a versão mais atual do código-fonte do caOS, compila-o, executa uma bateria de testes e envia os resultados para exibição no painel de controle (veja Seção 5.2 para detalhes). Os testes de unidade de unidade e integração são realizados sempre que uma alteração no código-fonte é detectada. Os testes de sistema incluem a instanciação e execução de vários fluxos de trabalho utilizando o caOS e demandam mais recursos computacionais. Por tal motivo, testes de sistema são realizados apenas uma vez por dia. À semelhança dos demais testes, seus resultados são enviados para exibição no painel de controle.

## 5.4 Sumário

Neste capítulo, o ambiente em que o caOS foi desenvolvido foi descrito, incluindo aspectos gerenciais e operacionais. No capítulo seguinte, serão apresentados os experimentos realizados para analisar a eficiência do caOS.

# Capítulo 6

# Avaliação experimental

Neste capítulo, são apresentados experimentos realizados utilizando o caOS. Tais experimentos foram realizados com o objetivo de comparar o sistema implementado com outras formas de executar um fluxo de trabalho científico e também com o objetivo de obter informações detalhadas sobre a execução das tarefas realizadas pelo caOS para criação de um fluxo de trabalho. Houve uma tentativa de comparar uma execução utilizando o caOS com uma execução utilizando o Taverna [Oinn et al., 2004b] a fim de observar a diferença dos tempos de execução entre o caOS, que emprega coreografia para execução, e um sistema que utiliza orquestração. No entanto, foram encontradas dificuldades para executar o fluxo de trabalho escolhido através do Taverna e o experimento não pôde ser realizado.

O restante do capítulo é organizado como segue. A Seção 6.1 apresenta e discute os resultados do experimento que foi realizado com o objetivo de estimar a perda de desempenho observada ao executar um fluxo de trabalho científico utilizando o caOS. Em seguida, é apresentado o experimento realizado para determinar as operações mais custosas realizadas durante a criação de um fluxo de trabalho na Seção 6.2.

## 6.1 Perda de desempenho imposta pelo caOS

Um experimento foi realizado a fim de determinar a perda de desempenho que um fluxo de trabalho experimenta ao ser executado através do caOS. Para estimar tal perda, o tempo total da execução de um fluxo de trabalho utilizando o caOS foi comparado com o tempo da execução feita sem qualquer ferramenta de auxílio à criação e execução de fluxos de trabalho. Esta última forma de execução foi implementada através de uma aplicação Java que utiliza os clientes dos serviços componentes do fluxo de trabalho para orquestrá-lo, enviando os dados de entrada de cada estágio, recuperando seus dados de saída e realizando as transferências de dados entre os estágios.

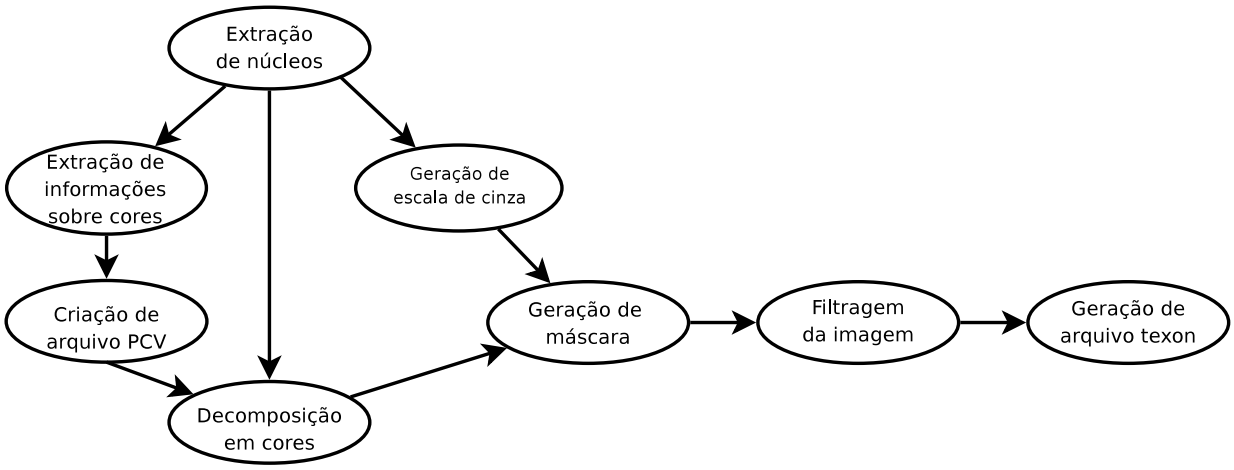


**Figura 6.1.** Diagrama dos estágios do TMA

O fluxo de trabalho utilizado nesse experimento é capaz de analisar imagens médicas e extrair características utilizando uma técnica conhecida como TMA [Chen et al., 2004] (do inglês *tissue microarray*). Tal fluxo de trabalho será chamado apenas de TMA daqui em diante. A técnica de diagnóstico citada corresponde a utilizar um conjunto de lâminas de tecidos e posicioná-las lado a lado numa só lâmina para análise simultânea durante um procedimento de biópsia. No TMA, uma imagem digitalizada desta última lâmina é analisada. As imagens das lâminas componentes da entrada do TMA são chamadas de núcleos. O TMA é composto por 8 estágios, a saber: (i) extração de núcleos, (ii) extração de informações sobre cores, (iii) construção de arquivo PCV, (iv) decomposição de cores, (v) geração da escala de cinza, (vi) geração de máscara, (vii) filtragem de imagem e (viii) geração de arquivo *texon*. O papel de cada estágio do TMA é descrito na Tabela 6.1 e um diagrama que representa o fluxo de trabalho é mostrado na Figura 6.1. Ocorre iteração em 2 estágios: geração de escala de cinza e decomposição de cores. Tais estágios recebem como entrada um arranjo de núcleos gerado pelo estágio de extração de núcleos e, como são executados uma vez para cada elemento do arranjo, os estágios seguintes no fluxo de trabalho também são executados múltiplas vezes.

Para esse experimento, foi utilizado um projeto fatorial completo com 2 fatores e 10 replicações. O número de replicações escolhido foi considerado adequado baseando-se no fato de que foram obtidas medidas com erros (largura do intervalo de confiança em relação à média) inferiores a 1.2%. Os fatores utilizados correspondem ao número de núcleos da imagem de entrada que o estágio de extração de núcleos retorna como saída e à forma de executar o fluxo de trabalho (utilizando caOS ou sem ferramenta de auxílio). Os valores utilizados para o número de núcleos foram 1, 2 e 10. Com tais valores foi possível observar a tendência do valor da perda de desempenho para execuções

| *Estágio* | *Tarefa realizada* |
|---|---|
| Extração de núcleos | Extrair os vários núcleos de uma única imagem |
| Extração de informações sobre cores | Extrair informações para decomposição de cores |
| Construção de arquivo PCV | Análise complementar para decomposição de cores |
| Decomposição em cores | Realizar decomposição de cores em imagens de núcleos |
| Geração de escala de cinza | Converter imagem colorida de um núcleo em imagem em escala de cinza |
| Geração de máscara | Criar máscara para filtragem de imagens |
| Filtragem de imagem | Aplicar filtro numa imagem de núcleo |
| Geração de arquivo *texon* | Extrair características de uma imagem de núcleo |

**Tabela 6.1.** Descrição de cada estágio do TMA

| *Experimento* | *Núcleos* | *Forma de execução* |
|---|---|---|
| 1 | 1 | com caOS |
| 2 | 2 | com caOS |
| 3 | 10 | com caOS |
| 4 | 1 | sem ferramenta de auxílio |
| 5 | 2 | sem ferramenta de auxílio |
| 6 | 10 | sem ferramenta de auxílio |

**Tabela 6.2.** Configuração dos experimentos realizados para estimar a perda de desempenho imposta pelo uso do caOS

que demandam um tempo próximo de uma hora. A configuração dos experimentos realizados é sumarizada na Tabela 6.2.

Os experimentos descritos nessa seção foram executados num computador com 8 processadores 2.6GHz AMD Opteron(tm) Dual-Core com 32GB de memória RAM. O sistema operacional presente no computador citado é Linux 2.6.18. Os serviços componentes do caOS e do fluxo de trabalho utilizado no experimento foram instalados num servidor Web Apache Tomcat 6.0.18.

Os resultados obtidos são mostrados na Tabela 6.3, incluindo o intervalo para o nível de confiança 90%. Examinando os tempos observados, é possível perceber que a execução do TMA através do caOS demandou um tempo maior para todos os valores de número de núcleos utilizados. Além disso, os intervalos de confiança para as médias não possuem interseção, o que mostra que existe uma perda de desempenho sempre que o TMA é executado utilizando o caOS. Ao calcular o acréscimo que a utilização do caOS impõe ao tempo demandado para executar o TMA em relação à execução sem ferramenta alguma de auxílio obtém-se um acréscimo de 15%, 5% e 3.4% para número de núcleos 1, 2 e 10, respectivamente. Esse resultado indica que a perda de desempenho imposta pelo caOS é pequena e tende a diminuir em relação ao tempo de execução. É importante ressaltar que uma execução completa do TMA utiliza aproximadamente 50 núcleos, o que não pôde ser feito no experimento devido a seu longo tempo de execução. Assim, o experimento é capaz de determinar a tendência das medidas realizadas para os números de núcleos escolhidos e seus resultados podem ser utilizados apenas como uma indicação do que seria observado para uma execução completa. No entanto, o experimento não revelou nenhum fator que poderia alterar a tendência observada.

| *Núcleos* | *Forma de execução* | *Tempo médio (s)* | *IC 90%* | *Erro %* |
|---|---|---|---|---|
| 1 | usando caOS | 391,4 | $(390,1; 392,8)$ | 0.71 |
| 1 | sem ferramenta de auxílio | 340,9 | $(340,7; 341,1)$ | 0.10 |
| 2 | usando caOS | 708 | $(705; 711)$ | 0.84 |
| 2 | sem ferramenta de auxílio | 672 | $(668; 676)$ | 1.14 |
| 10 | usando caOS | 3270 | $(3268; 3271)$ | 0.10 |
| 10 | sem ferramenta de auxílio | 3161 | $(3158; 3165)$ | 0.22 |

**Tabela 6.3.** Tempos observados para execução do TMA usando o caOS e sem o uso de ferramenta de auxílio à execução

O experimento mostra que a utilização do caOS pode ser uma alternativa vantajosa à criação e execução de fluxos de trabalho sem ferramenta alguma de auxílio por vários motivos. Primeiro, o tempo acrescentado à execução devido às tarefas realizadas pelo caOS para execuções com duração de alguns minutos é também de poucos minutos e corresponde a uma fração cada vez mais desprezível do tempo de execução para execuções mais longas. Segundo, a utilização do caOS não exige conhecimento profundo de programação da parte do usuário, exigindo apenas que este seja capaz de informar os serviços componentes do fluxo de trabalho e os dados que devem ser transferidos entre os mesmos. Ressalva-se aqui que o usuário deve compreender o uso da linguagem de consulta XPath [W3C, 2007c], conforme descrito na Seções 4.1.1 e 4.3.1. Por fim, o caOS não utiliza os clientes desenvolvidos pelos criadores dos serviços componentes do fluxo de trabalho para invocá-los mas é capaz de invocar serviços baseando-se apenas

em mensagens SOAP [W3C, 2007a]. Isso desobriga os criadores de serviços a distribuir os clientes para quem desejar utilizá-los em um fluxo de trabalho.

## 6.2 Custo computacional da criação de um fluxo de trabalho

Um segundo experimento foi realizado com o objetivo de determinar como o tempo observado para a criação de um fluxo de trabalho através do caOS está distribuído entre as diversas tarefas realizadas com tal finalidade. Em outras palavras, o experimento descrito nesta seção mostra quais tarefas são mais computacionalmente custosas durante a criação de um fluxo de trabalho.

Para realizar esse experimento, foi utilizado um pequeno fluxo de trabalho composto por 2 estágios apenas. Tal fluxo de trabalho é ilustrado na Figura 6.2. O primeiro estágio produz um arranjo com uma quantidade fixo de números inteiros como saída e o segundo estágio recebe um arranjo de números inteiros e apenas itera sobre seus elementos, sem realizar nenhum cálculo utilizando os mesmos. Tal fluxo de trabalho foi construído utilizando estágios cuja execução possui custo computacional desprezível a fim de que não fosse necessário incluir nas medidas realizadas os custos de execução de cada estágio. Em experimentos preliminares, foi determinado que o custo de execução dos estágios citados acima é da ordem de microssegundos e pode ser desprezado, pois todos as demais medidas realizadas possuem valores da ordem de milissegundos.

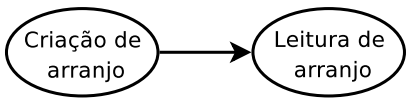


**Figura 6.2.** Diagrama do fluxo de trabalho utilizado no segundo experimento

O experimento foi realizado executando o fluxo de trabalho descrito acima com 15 replicações. Como as medidas realizadas apresentaram valores da ordem de milissegundos, considerou-se que erros (largura do intervalo de confiança em relação à média) inferiores a 12.2% são satisfatórios e o número de replicações escolhido é adequado. Foram medidos os tempos demandados pela criação e configuração de cada contexto que representa um elemento do fluxo de trabalho: instância de fluxo de trabalho, instância de fluxo de trabalho local (IFTL) e instância de estágio. A criação de um contexto é uma operação realizada pelo serviço correspondente, o que inclui a iniciação das estruturas de dados que fazem parte do contexto. A configuração de um contexto corresponde à criação de estruturas internas de dados e controle com base nas informações sobre o fluxo de trabalho em criação. A configuração de um contexto,

exceto para a instância de estágio, inclui também a criação e configuração de contextos pelos quais o mesmo é responsável. Assim, a configuração de uma instância de fluxo de trabalho inclui a criação e configuração de IFTLs. Por sua vez, a configuração de uma IFTL inclui a criação e configuração de instâncias de estágio. Também foram medidos os tempos gastos para delegar uma credencial e para obter uma credencial delegada junto ao CDS [Langella, 2007] conforme descrito na Seção 2.1. Por fim, foram medidos os tempos demandados para invocar um estágio de forma não-segura e de uma forma segura. Vale a pena ressaltar que as medidas dos tempos de invocação consistem no tempo demandado para criação da requisição SOAP [W3C, 2007a], seu envio e no armazenamento da mensagem SOAP que contém os dados de saída do estágio, pois o tempo de execução dos estágios é pequeno o suficiente para ser desprezado (da ordem de microssegundos).

Os tempos medidos para as diversas tarefas executadas durante a criação de um fluxo de trabalho são mostrados na Tabela 6.4, junto aos intervalos correspondentes ao nível de confiança de 90%. O tempo de configuração de uma instância de fluxo de trabalho é exatamente igual à soma dos tempos de criação e configuração da única IFTL do fluxo de trabalho utilizado no experimento, o que exemplifica a relação entre a configuração de um contexto e a criação e configuração de contextos pelos quais o mesmo é responsável. Duas instâncias de estágio foram criadas pelo caOS, mas o tempo observado para a configuração da IFTL não foi correspondente ao dobro da soma dos tempos de criação e configuração das instâncias de estágio. Isso se deve a uma tarefa adicional realizada durante a configuração da IFTL: o mapeamento das saídas do fluxo de trabalho em saídas de estágios individuais.

Observa-se também que o tempo de criação dos contextos são maiores para aqueles que representam elementos de mais baixo nível de abstração. Em outras palavras, o tempo de criação de uma instância de estágio é maior que o de criação de uma IFTL, que, por sua vez, é um tempo maior do que o observado para criação de uma instância de fluxo de trabalho. Isso ocorre porque as tarefas realizadas durante a criação dos contextos que representam elementos de mais baixo nível de abstração são mais complexas, incluindo, por exemplo, iniciação de um maior número de estruturas de controle. Além disso, informações mais detalhadas precisam ser armazenadas pelos contextos que representam elementos de mais baixo nível, o que contribui para que os tempos observados para configuração sejam maiores.

Também foram medidos os tempos demandados para a delegação de credenciais e a recuperação de credenciais delegadas junto ao CDS [Langella, 2007]. A delegação de credenciais é realizada pelo SG, que, após obter uma credencial delegada pelo usuário, precisa delegar tal credencial para as IFTLs responsáveis por estágios que necessitam

| *Operação* | *Tempo médio em 15 execuções (s)* | *Largura do IC 90% em relação à média* |
|---|---|---|
| Criação da instância de fluxo de trabalho | 0.02 | 12.12% |
| Configuração da instância de fluxo de trabalho | 2.94 | 1.77% |
| Criação da instância de fluxo de trabalho local | 0.32 | 2.47% |
| Configuração da instância de fluxo de trabalho local | 2.62 | 1.78% |
| Criação da instância de estágio | 0.70 | 2.43% |
| Configuração da instância de estágio | 0.32 | 3.11% |
| Delegação de credencial | 0.64 | 6.76% |
| Recuperação de credencial delegada | 0.45 | 6.14% |

**Tabela 6.4.** Tempo decorrido na execução das principais operações para criação de um fluxo de trabalho científico através do caOS.

da mesma para executar. A recuperação de credenciais delegadas ocorre, além do caso já citado, quando uma IFTL obtém uma credencial delegada por uma instância de fluxo de trabalho. O tempo medido para a delegação de uma credencial foi maior do que aquele medido para a recuperação de uma credencial delegada. Isso ocorre porque a delegação de uma credencial inclui a criação da credencial delegada, da qual faz parte um par de chaves criptográficas pública/privada. O tempo medido para delegação e recuperação de credenciais é comparável ao de criação dos contextos, o que indica que as operações envolvendo credenciais contribuem de forma importante para o custo de criação de um fluxo de trabalho.

Ainda nesse experimento, foram medidos os tempos observados para a invocação de uma operação de forma segura e para a invocação de uma operação de forma não-segura. Os tempos observados estão listados na Tabela 6.5. Pode-se observar que o tempo demandado para invocação de uma operação de forma segura é, aproximadamente, uma ordem de grandeza maior do que a invocação de uma operação de forma não-segura. Pode-se afirmar que os custos do estabelecimento da conexão segura e do constante uso de algoritmos de criptografia inerentes à invocação de uma operação de forma segura são os fatores que levam a um desempenho conforme o observado. Tal observação confirma a expectativa de que fluxos de trabalho que possuem, entre seus componentes, operações que devem ser invocadas de forma segura apresentarão desempenho pior se comparado àqueles compostos por estágios que devem ser invocados de forma não-

segura. É importante lembrar, porém, que o suporte ao acesso a serviços seguros não é encontrado em muitos dos sistemas de gerenciamento de fluxos de trabalho científicos existentes e é um dos principais objetivos deste trabalho. Assim sendo, o benefício trazido aos usuário do caOS torna-se mais importante do que a penalidade observada em cada execução de um serviço seguro, que é da ordem de décimos de segundo.

| *Tipo de invocação* | *Tempo médio em 15 execuções (s)* | *Largura do IC 90% em relação à média* |
|---|---|---|
| Segura e local | 0.12 | 9.69% |
| Não-segura e local | 0.02 | 21.07% |

**Tabela 6.5.** Tempo decorrido na invocação de uma operação. Foram medidos os tempos para invocações seguras e não-seguras.

É importante fazer 2 ressalvas quanto ao que foi discutido no parágrafo anterior. Primeiro, muitas vezes não é possível optar por invocar uma operação de forma segura ou não-segura, pois os requisitos de segurança para o acesso a um serviço são estabelecidos por seu criador e muitos serviços ou lidam com informações confidenciais, que não podem ser acessados de forma não-segura, ou lidam com informações disponíveis ao público em geral, caso em que não é necessário o acesso de forma segura. Segundo, não foi mostrado o tempo medido para a invocação de uma operação instalada num servidor Web hospedado em um computador diferente daquele em que a instância de estágio foi criada (operação remota), pois tal análise não resultou em nenhum conhecimento sobre o caOS, mas apenas expôs as características da rede utilizada para comunicação entre os elementos citados.

## 6.3 Sumário

Neste capítulo, os experimentos realizados para avaliar o caOS foram descritos e analisados em detalhes. O desempenho do caOS foi comparado à forma mais simples de criar e executar um fluxo de trabalho e foi analisado como a perda de desempenho associada ao uso do caOS está distribuída entre as tarefas que fazem parte do processo de criação e execução. No capítulo seguinte, serão apresentados as considerações finais sobre o trabalho e os trabalhos que poderão seguir o descrito nesta dissertação.

# Capítulo 7

# Conclusões

Neste capítulo, são apresentadas algumas considerações finais sobre o trabalho realizado, destacando os aspectos mais importantes. Tais considerações são encontradas na Seção 7.1. Além disso, a Seção 7.2 descreve vários trabalhos que podem ser realizados com base no apresentado nesta dissertação.

## 7.1 Considerações finais

O trabalho descrito nesta dissertação abordou a dificuldade de criação e execução de fluxos de trabalho científicos no caGrid por parte de pesquisadores que não são especialistas na área de Computação. Tal problema foi abordado através da construção de um sistema, o *caGrid Choreography System* (caOS), capaz de auxiliar na criação, execução e monitoramento de um fluxo de trabalho requerendo apenas uma descrição dos componentes do mesmo em formato XML.

A abordagem utilizada mostrou ser vantajosa ao simplificar o processo de desenvolvimento de um fluxo de trabalho provendo benefícios tanto ao usuário quanto aos desenvolvedores dos serviços componentes de um fluxo de trabalho científico. Para o usuário, o conhecimento técnico exigido foi restrito à criação da descrição do fluxo de trabalho, à capacidade de utilizar os métodos constantes da interface do Serviço de gerenciamento de fluxos de trabalho (SG) e à capacidade de delegar uma credencial ao caOS quando operações que requerem acesso seguro fazem parte do fluxo de trabalho. Para os desenvolvedores dos serviços, a necessidade de distribuir bibliotecas que implementam clientes para acesso aos serviços deixa de existir ao utilizar o caOS. É importante ressaltar também que requisitos específicos da comunidade do caGrid foram atendidos pelo caOS, como a capacidade de invocar um serviço de forma segura utilizando credenciais e o *caGrid Credential Delegation Service* [Langella, 2007] e a adoção de padrões como XPath [W3C, 2007c], SOAP [W3C, 2007a], WS-Notification [OASIS, 2006a] e

X.509 [ITU-T, 2008].

O desempenho do caOS apresentou perdas pequenas de desempenho ao executar um fluxo de trabalho que implementa uma aplicação real se comparado à forma mais simples de executá-lo. Também foi possível analisar a perda de desempenho experimentada ao utilizar-se o caOS determinando os custo das diversas tarefas realizadas pelo sistema para criar um fluxo de trabalho e executar um estágio individualmente.

A arquitetura desenvolvida durante o trabalho pode ser facilmente estendida e modificada, dando origem a outros trabalhos. Por ser baseada em serviços e possuir componentes desenvolvidos de forma modular, tal arquitetura pode ter novos componentes incluídos com poucas modificações na implementação existente. A adoção de padrões estabelecidos permite que o caOS seja capaz de criar fluxos de trabalho utilizando quaisquer serviços que obedeçam aos padrões SOAP [W3C, 2007a] e WSDL [W3C, 2007b]. Além disso, o trabalho foi desenvolvido como parte da incubadora de projetos do caGrid, o que o coloca como um trabalho que possui apoio da comunidade em torno do caGrid e continuará sendo desenvolvido a fim de atender às demandas de tal comunidade.

## 7.2 Trabalhos futuros

Nesta seção, serão descritos alguns trabalhos que podem dar continuidade àquele descrito nesta dissertação. Tais trabalhos incluem acréscimo de funcionalidades aos serviços componentes do caOS, desenvolvimento de novos componentes do caOS, integração com outras aplicações e extensão de outras aplicações. O objetivo das sugestões descritas a seguir é fornecer ideias para, além de tornar possível o prosseguimento do trabalho descrito aqui, estimular o desenvolvimento de outros trabalhos.

Uma funcionalidade que seria interessante acrescentar ao caOS é o suporte a cópias transparentes de estágios de forma semelhante ao que é implementado pelo Anthill [Tavares et al., 2007]. O Anthill é capaz de instanciar um mesmo estágio múltiplas vezes com a finalidade de balancear a carga de trabalho entre os mesmos. Assim, o Anthill é capaz de, dado um conjunto de dados de saída de execuções de um estágio, distribuir tais dados entre várias instâncias de um outro estágio e criar paralelismo de dados dentro do mesmo estágio. No caOS, tal característica poderia acelerar a execução de fluxos de trabalho contendo iterações. Dado um estágio que é executado múltiplas vezes devido à ocorrência de iteração num fluxo de trabalho e uma lista de descrições pelas quais é possível instanciar múltiplas instâncias do estágio, seria possível distribuir entre tais instâncias a responsabilidade de executar as iterações. Como as operações dos serviços Web não possuem estado, a execução das iterações é comutativa

e não haveria problemas devido a uma possível entrega desordenada dos resultados das iterações.

Outra funcionalidade que pode ser acrescentada ao caOS é o suporte aos serviços integráveis a fluxos de trabalho descritos na Seção 4.1.1. Tais serviços seriam parte de uma classe de serviços WSRF capazes de serem configurados como integrantes de um fluxo de trabalho, sendo invocados por uma entidade e enviando seus dados de saída para entidades para as quais foi configurada tal transferência. Dessa forma, a configuração de estágios seria simplificada no caOS. É necessário, entretanto, que haja um padrão a ser obedecido por tais serviços para que seja acrescentado suporte aos mesmos no caOS. Uma forma de alcançar esse objetivo seria desenvolver uma extensão para o Introduce [Hastings et al., 2007] que possibilita ao mesmo criar serviços integráveis a fluxos de trabalho. Assim, o processo de criação e utilização de serviços integráveis a fluxos de trabalho seria impulsionado.

Acrescentar ao caOS uma interface gráfica que permitisse o desenvolvimento de um fluxo de trabalho a partir de suas operações componentes é um dos principais requisitos para prosseguir em tornar sua utilização simples. Tal objetivo poderia ser alcançado integrando ao sistema uma interface gráfica já existente. Como a interface utilizada pelo Taverna [Oinn et al., 2004b] está em um estágio bastante maduro de desenvolvimento, a mesma poderia ser integrada ao caOS a fim de que fluxos de trabalho desenvolvidos a partir dela sejam enviados ao caOS para execução. Tal abordagem tem como principais benefícios: tornar desnecessário o desenvolvimento completo de uma interface gráfica com as funcionalidades necessárias; a familiaridade de diversos usuários com o Taverna, o que facilitaria o acesso dos mesmos ao caOS.

A criação de um novo componente que realize o papel de escalonador no caOS também é um trabalho que pode gerar resultados importantes. Tal escalonador teria como função principal escolher, dentre uma lista de possibilidades, onde os estágios de um fluxo de trabalho serão instanciados. A lista de possibilidades pode ser fornecida pelo usuário ou obtida por meio de consultas a um serviço de índice capaz de armazenar informações sobre serviços existentes e recuperar tais informações através de consultas. Além disso, a existência de um escalonador permitiria iniciar a implementação de mecanismos de tolerância a falhas no caOS.

Acrescentar proveniência aos dados de um fluxo de trabalho científico no caOS pode criar oportunidades de otimização de execuções. Ao conhecer a proveniência dos dados produzidos durante a execução de um fluxo de trabalho, poderia ser permitido ao usuário especificar dados que devem ser armazenados num serviço de armazenamento persistente a fim de reutilizar tais dados em execuções posteriores. Isso seria importante, por exemplo, durante a realização de experimentos do tipo varredura de

parâmetros nos quais parte dos dados produzidos pelo fluxo de trabalho não é alterada entre execuções. Além disso, conhecer a proveniência dos dados permitiria a realização de estudos detalhados sobre o fluxo dos dados em fluxos de trabalho científico, contribuindo para o desenvolvimento de tais fluxos de trabalho.

# Apêndice A

# Utilização do caOS

Este apêndice descreve os passos necessários para a utilização do caOS, o sistema produzido no trabalho descrito nessa dissertação. Serão abordados vários aspectos da utilização do caOS, desde sua instalação até a obtenção das saídas de um fluxo de trabalho científico. Espera-se que o texto apresentado a seguir forneça informações suficientes para que um usuário seja capaz de criar, executar e monitorar fluxos de trabalho através do caOS.

O restante deste apêndice está organizado como segue. A Seção A.1 apresenta os pré-requisitos para a utilização do caOS. A forma como os componentes do caOS são compilados e instalados é descrita na Seção A.2. A criação da descrição de um fluxo de trabalho é discutida na Seção A.3. Em seguida, a Seção A.4 descreve como instanciar e executar um fluxo de trabalho programaticamente, abordando também a forma de monitorar uma execução e obter dos dados de saída.

## A.1 Pré-requisitos para utilização

Nesta seção, serão apresentadas as dependências do caOS em relação a outros aplicativos e/ou bibliotecas. Também será mostrado como tais dependências devem ser utilizadas para que o caOS possa ser instalado. A lista de dependências inclui: o conjunto de bibliotecas para desenvolvimento de aplicações Java JDK, a ferramenta para auxílio à compilação Apache Ant, o conjunto de bibliotecas Globus WS-Core e o Apache Tomcat, que é um servidor Web.

O caOS foi desenvolvido utilizando a linguagem de programação Java e, por isso, sua compilação depende da existência de um JDK. O Apache Ant é utilizado para compilar todos os componentes do caOS utilizando um JDK existente. Além das dependências citadas, a compilação do caOS também depende das bibliotecas presentes no Globus WS-Core, pois muitas das funcionalidades do caOS baseiam-se em funcionalidades

providas por tal componente.

O caOS também possui dependências para, além de sua compilação, sua instalação e execução. Para instalação do caOS, é necessário haver um servidor Web Apache Tomcat. O conjunto de bibliotecas Globus WS-Core deve ser instalado em tal servidor Web antes da instalação do caOS. Para execução do caOS, são necessárias todas as dependências associadas à sua compilação e instalação, com exceção da ferramenta de auxílio à compilação Apache Ant e das bibliotecas da JDK, que podem ser substituídas pelas bibliotecas de uma JRE.

Além das dependências citadas acima, o caOS também utiliza o *caGrid Credential Delegation Service* (CDS) quando instancia um fluxo de trabalho que contém estágios que exigem o uso de credenciais, com o fim de delegar uma credencial para uma instância de fluxo de trabalho local (IFTL). Nesse caso, porém, não é necessário instalar o CDS em algum servidor Web, pois o caOS utiliza o mesmo CDS através do qual a credencial delegada pelo usuário foi obtida.

## A.2 Compilação e instalação

Esta seção descreve os processos de compilação e instalação do caOS. Assume-se aqui que todas as dependências citadas na Seção A.1 são satisfeitas.

O código-fonte do caOS pode ser obtido através do repositório do NCICB [1], no qual existe o projeto *caGrid Incubation*, do qual o caOS faz parte. A compilação do caOS é realizada através do Apache Ant. No momento da compilação, deve existir uma variável de ambiente indicando o diretório em que as bibliotecas do Globus WS-Core podem ser encontradas. A compilação gera um conjunto de bibliotecas Java que são usadas para a instalação do caOS.

A instalação também é feita com auxílio do Apache Ant, que instala as bibliotecas obtidas ao fim da compilação num servidor Apache Tomcat. Por isso, deve existir uma variável de ambiente indicando o diretório em que o Apache Tomcat está instalado, além da variável necessária durante a compilação. Cada componente do caOS possui processos de instalação um pouco diferentes. O serviço de gerenciamento de fluxos de trabalho (SG) pode ser instalado em apenas um computador, pois é capaz de gerenciar vários fluxos de trabalho simultaneamente. É necessário prover uma credencial para o SG, a fim de que um usuário seja capaz de delegar credenciais ao SG e este seja capaz de recuperá-las junto ao CDS. Caso exista um SG previamente instalado e acessível ao usuário, uma nova instalação é desnecessária e tal SG pode ser utilizado. O serviço de auxílio à execução (SE) deve ser instalado em todos computadores nos quais pretende-se

[1] http://gforge.nci.nih.gov

instanciar instâncias de estágio. Porém, melhor desempenho será observado caso o SE seja instalado em cada computador no qual exista uma operação que faz parte de um fluxo de trabalho que será executado, pois a comunicação local minimiza os custos de transferência de dados. Assim como o SG, uma credencial deve ser configurada como identificador do SE para que credenciais delegadas pelo SG possam ser recuperadas pelo SE junto ao CDS.

## A.3 Criação da descrição de um fluxo de trabalho

Nesta seção, é mostrado o processo de criação da descrição de um fluxo de trabalho científico a fim de que a mesma seja utilizada pelo caOS para instanciar um fluxo de trabalho. O SG possui apenas um método para instanciar fluxos de trabalho e tal método utiliza um objeto que deve construído contendo a descrição de um fluxo de trabalho científico. No entanto, as bibliotecas do caOS permitem ao usuário criar uma descrição de um fluxo de trabalho em formato XML e utilizá-la para, localmente, criar um objeto correspondente que pode ser utilizado para instanciar um fluxo de trabalho através do SG. A exigência de uma validação local de descrições em formato XML é importante pois permite que erros sejam detectados antes de invocar o SG.

A utilização de uma descrição em formato XML é opcional, ou seja, um objeto do tipo esperado pelo SG pode ser construído diretamente por um usuário. No entanto, a utilização de arquivos XML é mais simples e permite a construção de aplicações que realizem a interação necessária com o SG a fim de executar fluxos de trabalho. A descrição em formato XML deve conter as informações descritas na Seção 4.3.1 e deve obedecer ao formato definido num arquivo XSD que é disponibilizado em qualquer instalação do SG. Tal arquivo XSD pode ser utilizado em editores de arquivos XML para auxiliar a criação da descrição do fluxo de trabalho. É importante ressaltar que os valores dos parâmetros de entrada do fluxo de trabalho fazem parte de sua descrição e, por isso, devem ser incluídos na descrição em formato XML produzida. Uma vez que a descrição em formato XML é concluída, pode-se obter o objeto descritor correspondente através da biblioteca do caOS.

## A.4 Instanciação e execução

A instanciação de um fluxo de trabalho é realizada pelo caOS a partir de uma descrição do mesmo, criada conforme mostrado na Seção A.3. O método *criaInstânciaDeFluxo-DeTrabalho*, invocado tendo uma descrição como parâmetro, cria todos os componentes do fluxo de trabalho e tem como retorno um objeto através do qual é possível invocar

um método da instância de fluxo de trabalho correspondente para iniciar sua execução. Antes de iniciar a execução de um fluxo de trabalho, é necessário habilitar o recebimento de notificações de suas mudanças de estado a fim de poder determinar o término de uma execução. Tal operação será detalhada em outro parágrafo. Uma vez que a execução é concluída, uma notificação correspondente é enviada pelo caOS.

O monitoramento de uma execução possui 2 etapas. A primeira consiste em cadastrar o aplicativo através do qual o caOS foi contatado para que tal aplicativo receba notificações de mudanças no estado do fluxo de trabalho. Feito isso, o caOS informará cada mudança de estado ocorrida e tal notificação deverá ser interpretada pelo aplicativo que realizou o cadastro. Um método específico deve ser implementado para receber notificações, examinar seu conteúdo e realizar ações de acordo com o conteúdo da notificação. Por exemplo, caso seja recebida uma notificação de fim de execução, podem ser requisitados ao caOS os valores dos dados de saída correspondentes. As bibliotecas do caOS provêm ao usuário uma forma de simplificar o processo descrito, realizando as tarefas de cadastramento para receber notificações duma instância de fluxo de trabalho e também a tarefa de aguardar a notificação de fim de execução.

Após o término de uma execução, é possível recuperar os valores dos dados de saída do fluxo de trabalho correspondente. Tais valores são enviados pelo caOS como um arranjo no qual cada elemento corresponde a um dado de saída. Cada dado de saída é armazenado como texto XML, que pode ser utilizado para obter os objetos correspondentes através de bibliotecas de desserialização disponibilizadas por um JDK.

# Apêndice B

# Esquemas da descrição de um fluxo de trabalho científico

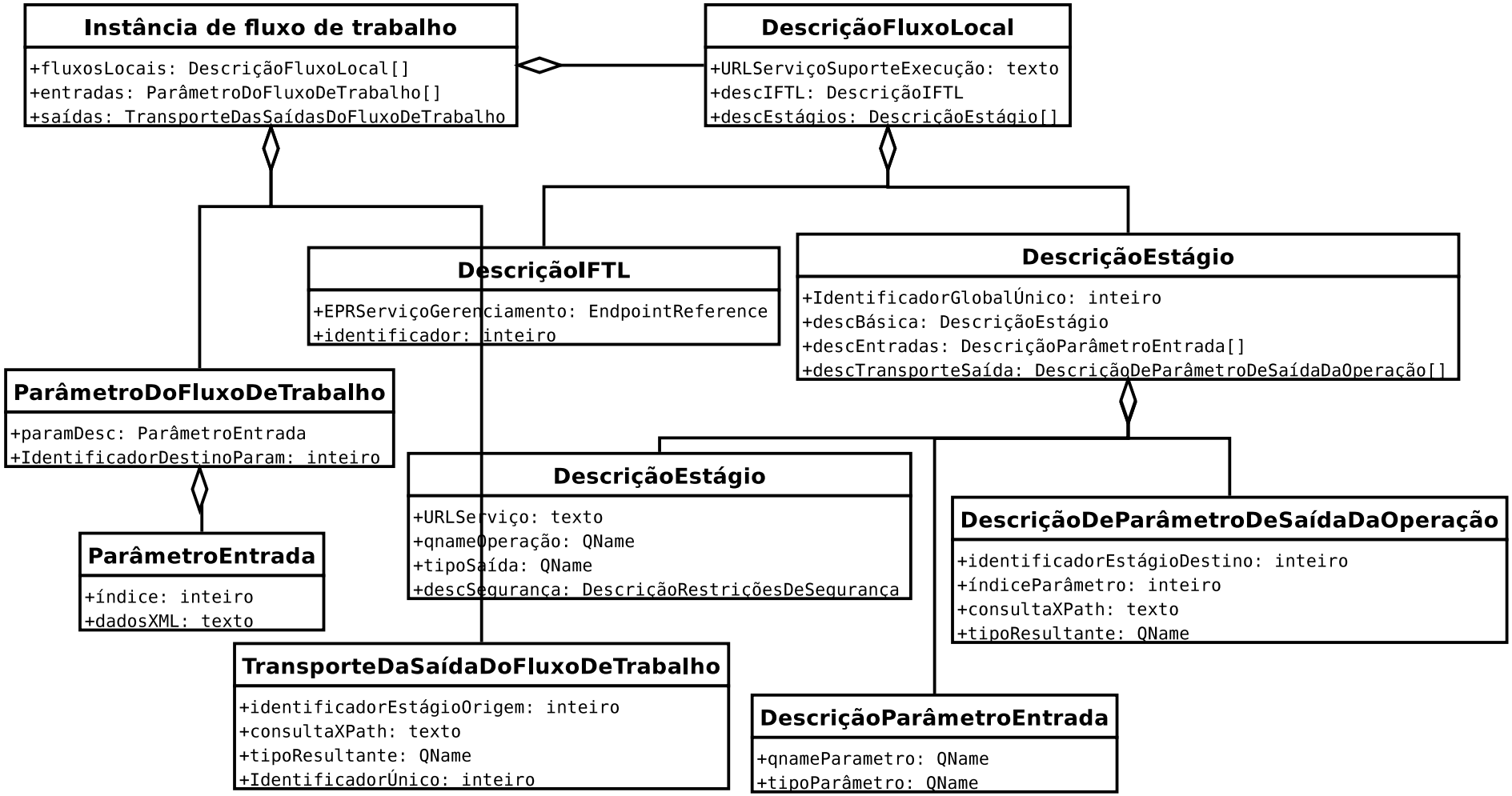


**Figura B.1.** Esquema da descrição de um fluxo de trabalho científico

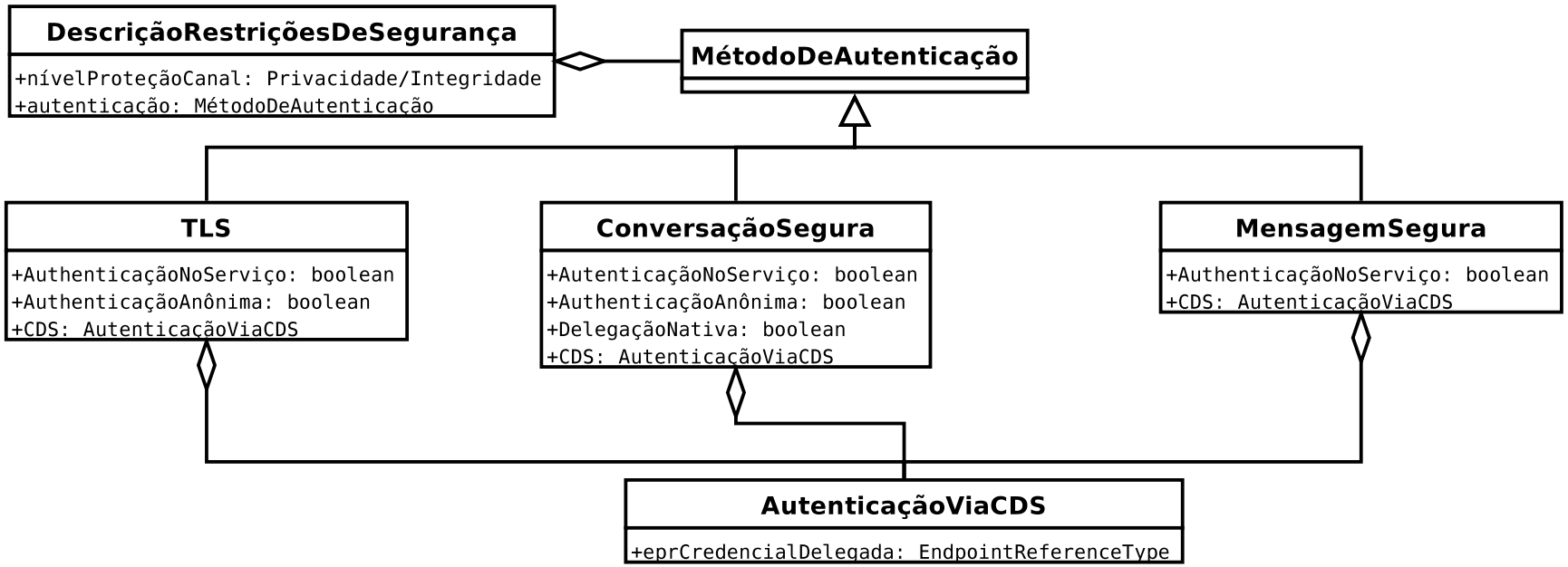


**Figura B.2.** Esquema da descrição das restrições de segurança de um estágio

# Referências Bibliográficas

[Allcock et al., 2002] Allcock, B.; Bester, J.; Bresnahan, J.; Chervenak, A. L.; Foster, I.; Kesselman, C.; Meder, S.; Nefedova, V.; Quesnel, D. e Tuecke, S. (2002). Data management and transfer in high-performance computational grid environments. *Parallel Comput.*, 28(5):749–771.

[Ardissono et al., 2007] Ardissono, L.; Furnari, R.; Goy, A.; Petrone, G. e Segnan, M. (2007). *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*. Springer Netherlands.

[Butler et al., 2000] Butler, R.; Welch, V.; Engert, D.; Foster, I.; Tuecke, S.; Volmer, J. e Kesselman, C. (2000). A national-scale authentication infrastructure. *Computer*, 33(12):60–66.

[caGrid, 2007] caGrid (2007). caGrid Query Language. http://cagrid.org/display/dataservices/CQL.

[caGrid, 2008] caGrid (2008). Incubação de projetos do caGrid. https://gforge.nci.nih.gov/projects/grid-incubation/.

[Cattlet, 2005] Cattlet, C. (2005). *TeraGrid: A Foundation for US Cyberinfrastructure*. Springer Berlin / Heidelberg.

[Chen et al., 2004] Chen, W.; Reiss, M. e Foran, D. (2004). A prototype for unsupervised analysis of tissue microarrays for cancer research and diagnostics. *Information Technology in Biomedicine, IEEE Transactions on*, 8(2):89–96.

[Covitz et al., 2003] Covitz, P. A.; Hartel, F.; Schaefer, C.; De Coronado, S.; Fragoso, G.; Sahni, H.; Gustafson, S. e Buetow, K. H. (2003). cacore: a common infrastructure for cancer informatics. *Bioinformatics (Oxford, England)*, 19(18):2404–2412.

[de Pádua Filho, 2009] de Pádua Filho, W. (2009). *Engenharia de Software: Fundamentos, Métodos e Padrões*. LTC.

[Deelman et al., 2004] Deelman, E.; Blythe, J.; Gil, Y. e Kesselman, C. (2004). Workflow management in griphyn. pp. 99–116.

[Deelman et al., 2005] Deelman, E.; Singh, G.; Su, M.-H.; Blythe, J.; Gil, Y.; Kesselman, C.; Mehta, G.; Vahi, K.; Berriman, G. B.; Good, J.; Laity, A. C.; Jacob, J. C. e Katz, D. S. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237.

[Feller et al., 2007] Feller, M.; Foster, I. e Martin, S. (2007). GT4 GRAM: A Functionality and Performance Study. *TeraGrid*.

[Ferreira et al., 2005] Ferreira, R.; Jr., W. M.; Guedes, D.; Drummond, L.; Coutinho, B.; Teodoro, G.; Tavares, T.; Araujo, R. e Ferreira, G. (2005). Anthill: a scalable run-time environment for data mining applications. pp. 159–166.

[Foster, 2005] Foster, I. T. (2005). Globus toolkit version 4: Software for service-oriented systems. In Jin, H.; Reed, D. A. e Jiang, W., editores, *NPC*, volume 3779 of *Lecture Notes in Computer Science*, pp. 2–13. Springer.

[Foster et al., 1998] Foster, I. T.; Kesselman, C.; Tsudik, G. e Tuecke, S. (1998). A security architecture for computational grids. In *ACM Conference on Computer and Communications Security*, pp. 83–92.

[Gamma et al., 1994] Gamma, E.; Helm, R.; Johnson, R. e Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

[Geddes, 2006] Geddes, N. (2006). The national grid service of the uk. *e-Science and Grid Computing, International Conference on*, 0:94.

[Gropp et al., 1996] Gropp, W.; Lusk, E. e Skjellum, A. (1996). Using mpi-portable parallel programming with the message-passing interface. *Sci. Program.*, 5(3):275–276.

[Hastings, 2008] Hastings, S. (2008). caGrid Transfer Service. http://www.cagrid.org.

[Hastings et al., 2007] Hastings, S.; Oster, S.; Langella, S.; Ervin, D.; Kurç, T. M. e Saltz, J. H. (2007). Introduce: An open source toolkit for rapid development of strongly typed grid services. *J. Grid Comput.*, 5(4):407–427.

[Henderson, 1995] Henderson, R. L. (1995). Job scheduling under the portable batch system. In *IPPS '95: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 279–294, London, UK. Springer-Verlag.

[ITU-T, 2008] ITU-T (2008). Itu-t x.509 standard. http://www.itu.int/rec/T-REC-X.509.

[Kakazu et al., 2004] Kakazu, K.; Cheung, L. W. e Lynne, W. (2004). The cancer biomedical informatics grid (cabig): Pioneering an expansive network of information and tools for collaborative cancer research. *Hawaii Medical Journal*, 63(9):273–276.

[Karonis et al., 2003] Karonis, N. T.; Toonen, B. e Foster, I. (2003). Mpich-g2: a grid-enabled implementation of the message passing interface. *J. Parallel Distrib. Comput.*, 63(5):551–563.

[Kesselman e Foster, 1998] Kesselman, C. e Foster, I. (1998). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers.

[Konagaya, 2002] Konagaya, A. (2002). Obigrid: Towards a new distributed platform for bioinformatics. In *SRDS '02: Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, p. 380, Washington, DC, USA. IEEE Computer Society.

[Langella, 2007] Langella, S. (2007). caGrid Credential Delegation Service. http://www.cagrid.org.

[Langella et al., 2006] Langella, S.; Oster, S.; Hastings, S.; Siebenlist, F.; Kurc, T. e Saltz, J. (2006). Dorian: Grid service infrastructure for identity management and federation. In *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, pp. 756–761.

[Langella et al., 2007] Langella, S.; Oster, S.; Hastings, S. L.; Siebenlist, F.; Phillips, J.; Ervin, D. W.; Permar, J. D.; Kurc, T. M. e Saltz, J. H. (2007). The cancer biomedical informatics grid (cabig) security infrastructure. *Proceedings of the 2007 AMIA Annual Symposium.*

[Laure et al., 2006] Laure, E.; Fisher, S. M.; Fronher, A.; Grandi, C.; Kunszt, P. e Krenk, A. (2006). Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33–45.

[Mandal et al., 2007] Mandal, N.; Deelman, E.; Mehta, G.; Su, M.-H. e Vahi, K. (2007). Integrating existing scientific workflow systems: the kepler/pegasus example. In *WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science*, pp. 21–28, New York, NY, USA. ACM.

[Nelson, 1981] Nelson, B. J. (1981). *Remote procedure call.* PhD thesis, Pittsburgh, PA, USA.

[OASIS, 2006a] OASIS (2006a). Oasis Web Services Notification. http://www.oasis-open.org/committees/wsn/.

[OASIS, 2006b] OASIS (2006b). OASIS Web Services Resource Framework. http://www.oasis-open.org/committees/wsrf/.

[OASIS, 2006c] OASIS (2006c). Oasis Web Services Security. http://www.oasis-open.org/committees/wss/.

[OASIS, 2007] OASIS (2007). OASIS Web Services Business Process Execution Language. http://www.oasis-open.org/committees/wsbpel.

[Oinn et al., 2004a] Oinn, T.; Addis, M.; Ferris, J.; Marvin, D.; Greenwood, M.; Goble, C.; Wipat, A.; Li, P. e Carver, T. (2004a). Delivering web service coordination capability to users. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pp. 438–439, New York, NY, USA. ACM.

[Oinn et al., 2004b] Oinn, T.; Addis, M.; Ferris, J.; Marvin, D.; Senger, M.; Greenwood, M.; Carver, T.; Glover, K.; Pocock, M. R.; Wipat, A. e Li, P. (2004b). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054.

[Oster e Saltz, 2004] Oster, S. H. S. L. S. e Saltz, J. (2004). Distributed data management and integration: The mobius project. In *GGF Semantic Grid Workshop 2004*, pp. 20–38. GGF.

[Peltz, 2003] Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10):46–52.

[Saltz et al., 2006] Saltz, J.; Oster, S.; Hastings, S.; Langella, S.; Kurc, T.; Sanchez, W.; Kher, M.; Manisundaram, A.; Shanbhag, K. e Covitz, P. (2006). cagrid: design and implementation of the core architecture of the cancer biomedical informatics grid. *Bioinformatics*, 22(15):1910–1916.

[Stevens et al., 2003] Stevens, R. D.; Robinson, A. J. e Goble, C. A. (2003). mygrid: personalised bioinformatics on the information grid. *Bioinformatics*, 19 Suppl 1.

[Tan et al., 2008] Tan, W.; Foster, I. e Madduri, R. (2008). Combining the power of taverna and cagrid: Scientific workflows that enable web-scale collaboration. *IEEE Internet Computing*, 12(6):61–68.

[Tavares et al., 2007] Tavares, T.; Teodoro, G.; Kurc, T.; Ferreira, R.; Guedes, D.; Meira, W. J.; Catalyurek, U.; Hastings, S.; Oster, S.; Langella, S. e Saltz, J. (2007). An efficient and reliable scientific workflow system. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:445–452.

[Taylor et al., 2003] Taylor, I.; Shields, M.; Wang, I. e Philp, R. (2003). Grid enabling applications using triana.

[van der Aalst e van Hee, 2002] van der Aalst, W. e van Hee, K. (2002). *Workflow Management : Models, Methods, and Systems (Cooperative Information Systems)*. The MIT Press.

[W3C, 2005] W3C (2005). Web Services Choreography Description Language. http://www.w3.org/TR/ws-cdl-10/.

[W3C, 2007a] W3C (2007a). W3c Simple Object Access Protocol 1.2. http://www.w3.org/TR/soap12/.

[W3C, 2007b] W3C (2007b). W3c Web Services Description Language 2.0. http://www.w3.org/TR/wsdl20/.

[W3C, 2007c] W3C (2007c). XML Path Language. http://www.w3.org/TR/xpath20/.

[W3C, 2008] W3C (2008). Extensible markup language (xml). http://www.w3.org/TR/xml/.